

## Практическая работа №5 «Медиа-запросы в CSS. Flexible Box Layout, свойства flex-контейнеров и flex-элементов»

### 9 Медиа-запросы (@media)

Медиа-запрос (@media) в CSS используется для создания адаптивного веб-дизайна. Это означает, что представление веб-страницы отличается от системы к системе в зависимости от типа экрана или мультимедиа. Точка останова (*breakpoint*) указывает, при каком размере ширины устройства содержимое только начинает меняться.

Медиа-запросы можно использовать для проверки многих вещей:

- Ширины и высоты области просмотра (*viewport*)
- Ширины и высоты устройства
- Ориентации
- Разрешения

При вёрстке адаптивных сайтов часто нужно, чтобы какой-то набор стилей применялся только при соблюдении определённых условий. Например, текст должен стать зелёным только при горизонтальной ориентации смартфона. Или блок займёт всю ширину родителя, если ширина экрана больше или равна 1500 пикселям. Для подобных случаев и существуют медиавыражения. Они помогают адаптировать вёрстку под разные экраны и устройства и при этом не писать лишний код.

Пример синтаксиса (листинги 9.1.1, 9.1.2):

Листинг 9.1.1 – Пример синтаксиса медиа-запроса в общем виде

```
@media [тип устройства] [характеристика устройства] {  
    /* CSS-правила */  
}
```

Листинг 9.1.2 – Пример синтаксиса медиа-запроса в общем виде

```
@media print and (orientation: landscape) {  
    .block {  
        font-size: 25pt;  
    }  
}
```

В листинге 9.1.2 использован тип устройства `print` (принтер), а характеристика устройства — `orientation: landscape` (альбомная ориентация). То есть при печати на листе в альбомной ориентации размер шрифта у блока будет 25 пунктов.

Вы можете вышепредставленным образом использовать `@media` в своей таблице стилей, или можете создать отдельную таблицу стилей и использовать атрибут `media` в элементе ссылки (листинг 9.1.3):

Листинг 9.1.3 – Альтернативное добавление описания медиа-запроса для `print`

```
<link rel="stylesheet" href="landscape.css" media="print and (orientation: landscape)">
```

**Примечание:** при использовании типа устройства `print` (принтер) для вывода на печать рекомендуется применять следующие единицы измерения – `em`, `cm`, `mm`, `in`, `pt`, `pc`, `%`.

Есть **три типа устройств**, которые можно использовать для адаптации с помощью медиа-запросов:

- `all` — медиавыражение применится ко всем устройствам. Если не задать никакой тип, по умолчанию применится этот.
- `print` — стили внутри такого медиавыражения применятся при печати на принтерах или экспорте в PDF, в том числе в режиме предпросмотра документа. Так, например, вы можете захотеть, чтобы ваши веб-страницы выглядели одним образом на экране, но по-другому при печати.

В примере ниже (листинг 9.1.4) цвет фона установлен на серый. Но если страница печатается, цвет фона должен быть прозрачным. Это экономит чернила принтера пользователя.

Листинг 9.1.4 – Применение медиа-запроса для принтера

```
body {
  color: black;
  background-color: grey;
}

@media print {
  body {
    background-color: transparent;
  }
}
```

- `screen` — для устройств с экранами.

**Примечание:** в большинстве случаев, когда вы пишете стили только для экрана, указывать типы `screen` или `all` не нужно. Реальное практическое использование есть только у типа `print`.

## 9.1 Категории характеристик устройств для использования медиа-запросов

Описание характеристик страницы и окна браузера представлено в таблице 9.2.1.

Таблица 9.2.1 - Характеристики страницы и окна браузера

Характеристика	Описание
<code>width</code>	Ширина окна браузера.
<code>height</code>	Высота окна браузера.
<code>min-width</code>	Минимальная ширина окна браузера.
<code>max-width</code>	Максимальная ширина окна браузера.
<code>min-height</code>	Минимальная высота окна браузера.
<code>max-height</code>	Максимальная высота окна браузера.
<code>aspect-ratio</code>	Соотношение между шириной и высотой окна.
<code>min-aspect-ratio</code>	Минимальное соотношение между шириной и высотой окна.
<code>max-aspect-ratio</code>	Максимальное соотношение между шириной и высотой окна.
<code>orientation</code>	Ориентация окна браузера: <code>landscape</code> (альбомная, горизонтальная) или <code>portrait</code> (портретная, вертикальная).
<code>overflow-block</code>	Проверка, как устройство вывода обрабатывает содержимое, которое выходит за пределы области просмотра по оси блока.
<code>overflow-inline</code>	Проверка, можно ли прокручивать содержимое, выходящее за пределы области просмотра по встроенной оси.

**Примечание:** вы можете использовать любые единицы длины CSS в своих медиа-запросах. Если ваш контент в основном основан на **изображениях**, **пиксели** имеют наибольший смысл для использования в данном случае.

Если же ваш контент в основном основан на **тексте**, имеет смысл использовать **относительную единицу**, основанную на размере текста, например, **em**.

Ниже приведен пример использования медиа-запроса с `orientation` окна браузера `landscape` и `portrait` (листинг 9.2.1, рисунки 9.2.1 и 9.2.2).

Листинг 9.2.1 – Медиа-запрос относительно ориентации окна браузера

```
@media screen and (orientation: landscape) {  
  body::after {  
    content: "Landscape";  
  }  
}  
  
@media screen and (orientation: portrait) {  
  body::after {  
    content: "Portrait";  
  }  
}  
  
body {  
  margin: 0;  
  display: flex;  
  align-items: center;  
  justify-content: center;  
  height: 100vh;  
  width: 100vw;  
  font-size: 10vmax;  
}
```

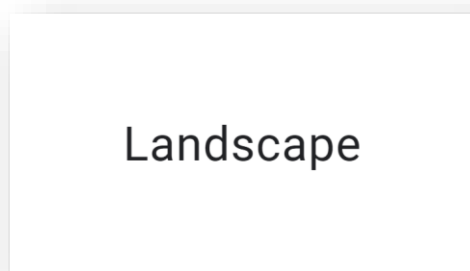


Рисунок 9.2.1 – Альбомная ориентация окна браузера

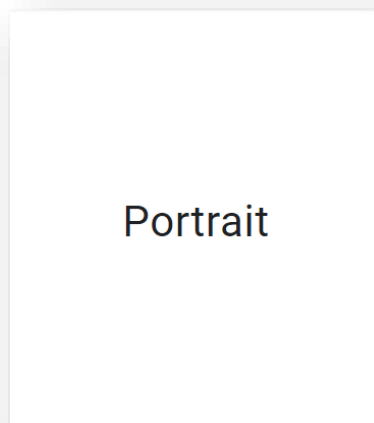


Рисунок 9.2.1 – Портретная ориентация окна браузера

Описание характеристик качества изображения представлено в таблице 9.2.2.

Таблица 9.2.2 - Характеристики качества изображения

Характеристика	Описание
<code>environment-blending</code>	Метод для определения внешнего окружения устройства, такого как тусклое или слишком яркое освещение.
<code>display-mode</code>	Проверка режима браузера, используется в PWA: <code>fullscreen</code> (полноэкранный режим без интерфейса браузера), <code>standalone</code> (как нативное приложение), <code>minimal-ui</code> (минимальный интерфейс браузера) и <code>browser</code> (обычное окно браузера).
<code>grid</code>	Проверка, является ли экран растровым (все современные экраны) или сеточным (как старые телефоны или текстовые терминалы).
<code>resolution</code>	Разрешение устройства в dpr или dpcm.
<code>min-resolution</code>	Минимальное разрешение устройства в dpr или dpcm.
<code>max-resolution</code>	Максимальное разрешение устройства в dpr или dpcm.
<code>scan</code>	Процесс сканирования устройства вывода.
<code>update</code>	Скорость обновления экрана: <code>none</code> (не обновляется), <code>slow</code> (медленно), <code>fast</code> (быстро).

Описание характеристик для цвета представлено в таблице 9.2.3.

Таблица 9.2.3 - Характеристики цвета

Характеристика	Описание
<code>color</code>	Количество бит на цвет на устройстве вывода.
<code>min-color</code>	Минимальное количество бит на цвет на устройстве вывода.
<code>max-color</code>	Максимальное количество бит на цвет на устройстве вывода.
<code>color-index</code>	Количество цветов, которое может отображаться устройством.
<code>min-color-index</code>	Минимальное количество цветов, которое может отображаться устройством.
<code>max-color-index</code>	Максимальное количество цветов, которое может отображаться устройством.
<code>monochrome</code>	Количество бит на цвет на монохромном устройстве вывода.
<code>min-monochrome</code>	Минимальное количество бит на цвет на монохромном устройстве вывода.
<code>max-monochrome</code>	Максимальное количество бит на цвет на монохромном устройстве вывода.
<code>color-gamut</code>	Примерный диапазон цветов, поддерживаемый браузером и устройством вывода.
<code>dynamic-range</code>	Комбинация уровня яркости, глубины цвета и контрастного соотношения для видео в браузере или устройстве вывода.
<code>inverted-colors</code>	Проверка, инвертируются ли цвета браузером или ОС.

**Примечание:** также существуют характеристики для проверки, включены ли скрипты – `scripting`, и для пользовательских настроек.

## 9.2 Ключевые слова (`and`, `not`, `only`) с несколькими условиями в медиа-запросах

В медиавыражении может быть перечислено несколько условий через запятую. В таком случае стили применяются для любого из указанных условий. Например (листинг 9.3.1):

Листинг 9.3.1 – Несколько условий в медиа-запросе

```
@media (orientation: landscape), (max-width: 960px) {  
  .text {  
    color: green;  
  }  
}
```

Такой код применится на всех устройствах с альбомной ориентацией экрана **или** на всех устройствах с шириной экрана менее 960 пикселей.

Часто в медиавыражениях может использоваться ключевое слово `and`, реже встречаются ключевые слова `not` и `only`.

1. Если мы пишем **и тип устройства, и указываем характеристики**, то после типа обязательно пишется `and` (листинг 9.3.2).

Листинг 9.3.2 – Использование ключевого слова `and` в медиа-запросе

```
@media print and (min-width: 320px) {  
  .link {  
    text-decoration: underline;  
  }  
}
```

Если мы **указываем несколько характеристик**, то между ними также пишется `and` (листинг 9.3.3).

Листинг 9.3.3 - Использование ключевого слова `and` в медиа-запросе

```
@media (min-width: 320px) and (max-width: 640px) {  
  .link {  
    text-decoration: underline;  
  }  
}
```

2. Ключевое слово `not` используется для **отрицания выражения**. Оно имеет низкий приоритет и применяется в последнюю очередь. Например (листинг 9.3.4):

Листинг 9.3.4 - Использование ключевого слова `not` в медиа-запросе

```
@media not screen and (min-width: 380px) {  
  .block {  
    display: flex;  
  }  
}
```

Также ключевое слово `not` используется для **инвертирования**, т.е. как в примере ниже (листинг 9.3.5), сначала вычислится выражение «для всех экранов с минимальной шириной 320 пикселей», а потом оно инвертируется — «для всех устройств, кроме устройств с экранами с минимальной шириной 320 пикселей».

Листинг 9.3.5 - Использование ключевого слова `not` в медиа-запросе

```
@media not screen and (min-width: 320px) {  
  .block {  
    display: flex;  
  }  
}
```

При использовании `not` в медиавыражениях с запятой инвертируется только та часть выражения, где указан `not`. Например (листинг 9.3.6):

Листинг 9.3.6 - Использование ключевого слова `not` в медиа-запросе

```
@media not screen and (height: 500px), print and (height: 700px) {  
  .text {  
    color: green;  
  }  
}
```

3. Ключевое слово `only` иногда встречается перед указанием типа устройства — оно используется для того, чтобы старые браузеры, которые не поддерживают медиавыражения, не применяли указанные стили. Современные браузеры никак не реагируют на `only` (листинг 9.3.7).

Листинг 9.3.7 - Использование ключевого слова `only` в медиа-запросе

```
@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

### 9.3 Выбор точки останова (*breakpoint*) на основе содержимого

Точка, в которой условие функции мультимедиа становится истинным, называется точкой останова. Лучше выбирать точки останова на основе вашего контента, а не популярных размеров устройств, поскольку они могут меняться с каждым циклом выпуска технологии.

В примере ниже (листинг 9.4.1) *50em* — это точка, в которой строки текста становятся неудобно длинными. Поэтому создается точка останова, чтобы сделать интерфейс более разборчивым. Используя свойство `column-count`, текст с этого момента делится на две колонки.

Листинг 9.4.1 – Выбор точки останова для разделения статьи на колонки

```
@media (min-width: 50em) {  
  article {  
    column-count: 2;  
  }  
}
```



## 10 Flexible Box Layout (Flexbox)

Модуль Flexible Box, или просто «flexbox» для краткости, представляет собой блочную модель, предназначенную для пользовательских интерфейсов, и позволяет пользователям выравнивать и распределять пространство между элементами в контейнере таким образом, чтобы элементы вели себя предсказуемо, когда макет страницы должен соответствовать различным, неизвестным размерам экрана. Flex-контейнер расширяет элементы, чтобы заполнить доступное пространство, и сжимает их, чтобы предотвратить переполнение.

Основная идея флексов — гибкое распределение места между элементами, гибкая расстановка, выравнивание, гибкое управление. Ключевое слово — гибкое, что и отражено в названии (flex — *англ.* гибко).

*Прежде чем применять какой-либо макет, вы должны убедиться, что поток вашего контента имеет смысл (листинг 10.1.1). Этот порядок по умолчанию с одним столбцом — это то, что будет отображаться на меньших экранах (рисунок 10.1.1).*

Листинг 10.1.1 – Структура смыслового потока контента

```
<body>
  <header>...</header>
  <main>
    <article>...</article>
    <aside>...</aside>
  </main>
  <footer>...</footer>
</body>
```

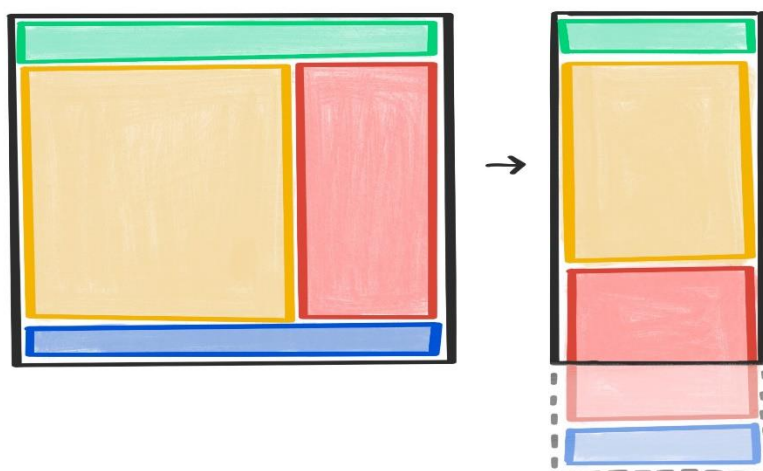


Рисунок 10.1.1 – Пример отображения контента на мобильной версии устройства

## 10.1 Основные понятия flexbox

1. **Flex-контейнер** – элемент, к которому применяется свойство `display: flex`. Вложенные в него элементы подчиняются правилам раскладки флексов (рисунок 10.2.1).
2. **Flex-элемент** – элемент, вложенный во flex-контейнер (рисунок 10.2.1).

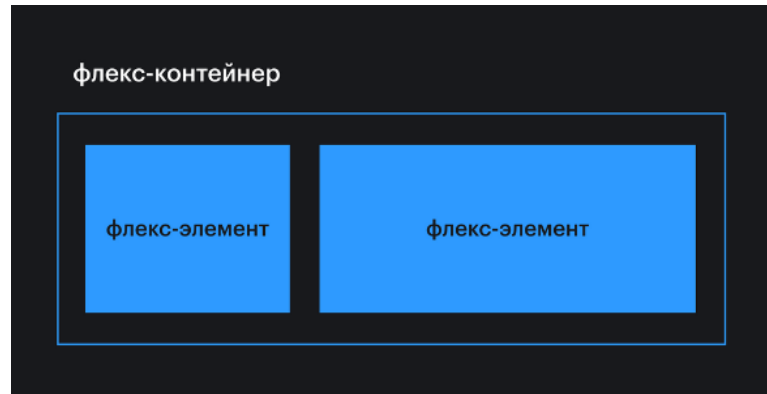


Рисунок 10.2.1 – Представление flex-контейнера

3. **Основная ось** – основная направляющая flex-контейнера, вдоль которой располагаются flex-элементы (рисунок 10.2.2).
4. **Поперечная (побочная, перпендикулярная) ось** – ось, идущая перпендикулярно основной (рисунок 10.2.2).

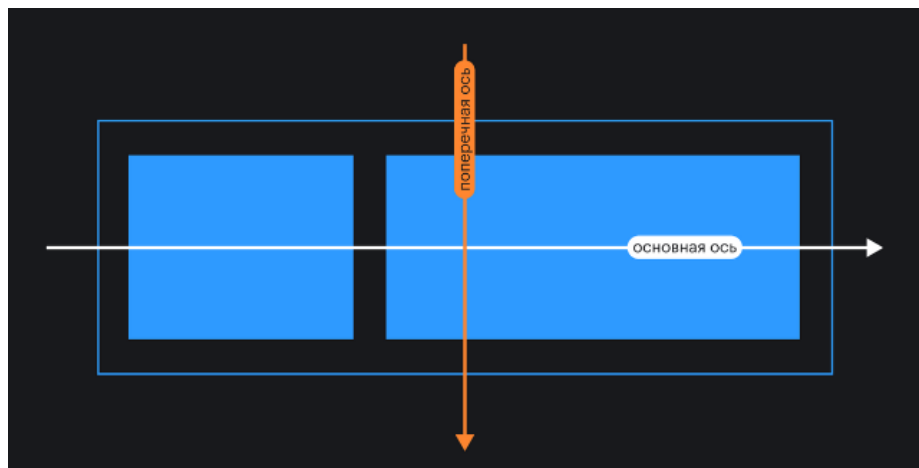


Рисунок 10.2.2 – Оси flex-контейнера

5. **Начало / конец основной оси** – точки в начале и в конце основной оси соответственно. Это пригодится для выравнивания flex-элементов (рисунок 10.2.3).
6. **Начало / конец поперечной оси** – точки в начале и в конце поперечной оси соответственно (рисунок 10.2.3).

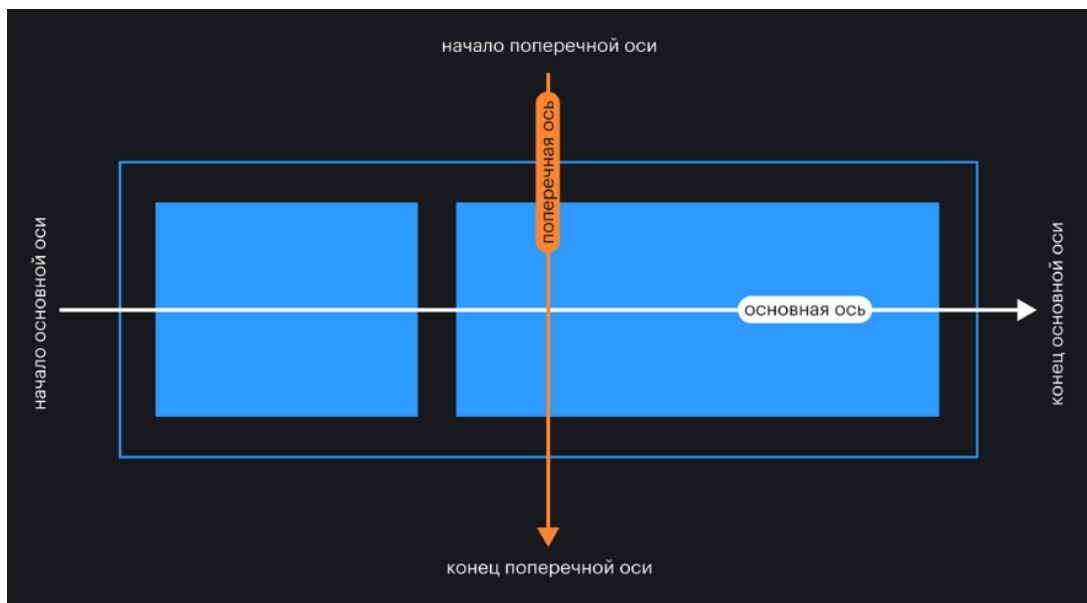


Рисунок 10.2.3 – Начало и концы осей flex-контейнера

**7. Размер по основной оси (основной размер)** - размер flex-элемента вдоль основной оси. Это может быть ширина или высота в зависимости от направления основной оси (рисунок 10.2.4).

**8. Размер по поперечной оси (поперечный размер)** - размер flex-элемента вдоль поперечной оси. Это может быть ширина или высота в зависимости от направления поперечной оси. Этот размер всегда перпендикулярен основному размеру. Если основной размер — это ширина, то поперечный размер — это высота, и наоборот (рисунок 10.2.4).

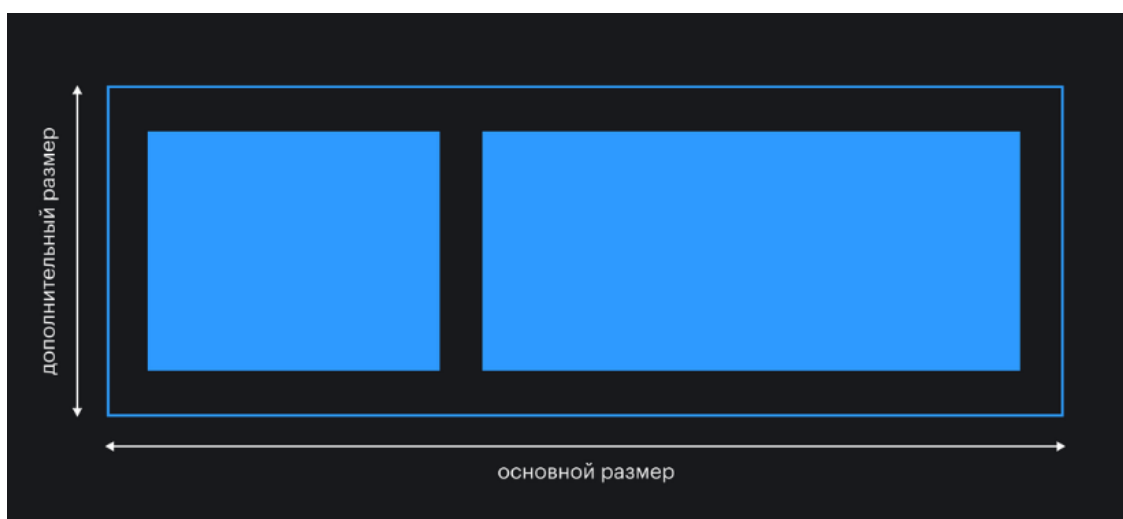


Рисунок 10.2.4 – Размеры flex-элементов по осям

## 10.2 Свойства flex-контейнера

### 10.2.1 Свойство display

Если задать какому-то элементу свойство `display` со значением `flex`, то этот элемент превратится во flex-контейнер (листинг 10.3.1). Внутри него начинает действовать flex-контекст, его дочерние элементы начинают подчиняться свойствам flexbox.

Снаружи flex-контейнер выглядит как блочный элемент — занимает всю ширину родителя, следующие за ним элементы в разметке переносятся на новую строку.

Листинг 10.3.1 – Создание flex-контейнера

```
.container {  
  display: flex;  
}
```

Если контейнеру задано свойство `display` со значением `inline-flex`, то снаружи он начинает вести себя как строчный элемент — встаёт в строку с другими элементами и размеры зависят только от внутреннего контента, а внутри это такой же flex-контейнер, как и при значении `flex`.

### 10.2.2 Свойство flex-direction

Данное свойство предназначено для управления направлением основной и поперечной осей.

В таблице 10.3.1 обозначены значения свойства `flex-direction`.

Таблица 10.3.1 - Значения свойства flex-direction

Значение свойства <code>flex-direction</code>	Описание значения
<code>row</code>	Значение по умолчанию — основная ось идёт горизонтально слева направо, поперечная ось идёт вертикально сверху вниз.
<code>row-reverse</code>	Основная ось идёт горизонтально справа налево, поперечная ось идёт вертикально сверху вниз.
<code>column</code>	Основная ось идёт вертикально сверху вниз, поперечная ось идёт горизонтально слева направо.
<code>column-reverse</code>	Основная ось идёт вертикально снизу вверх, поперечная ось идёт горизонтально слева направо.

Представление в коде (листинг 10.3.2) и визуально (рисунки 10.3.1, 10.3.2):

Листинг 10.3.2 – Свойство flex-direction

```
.container {  
  display: flex;  
  flex-direction: row;  
}
```

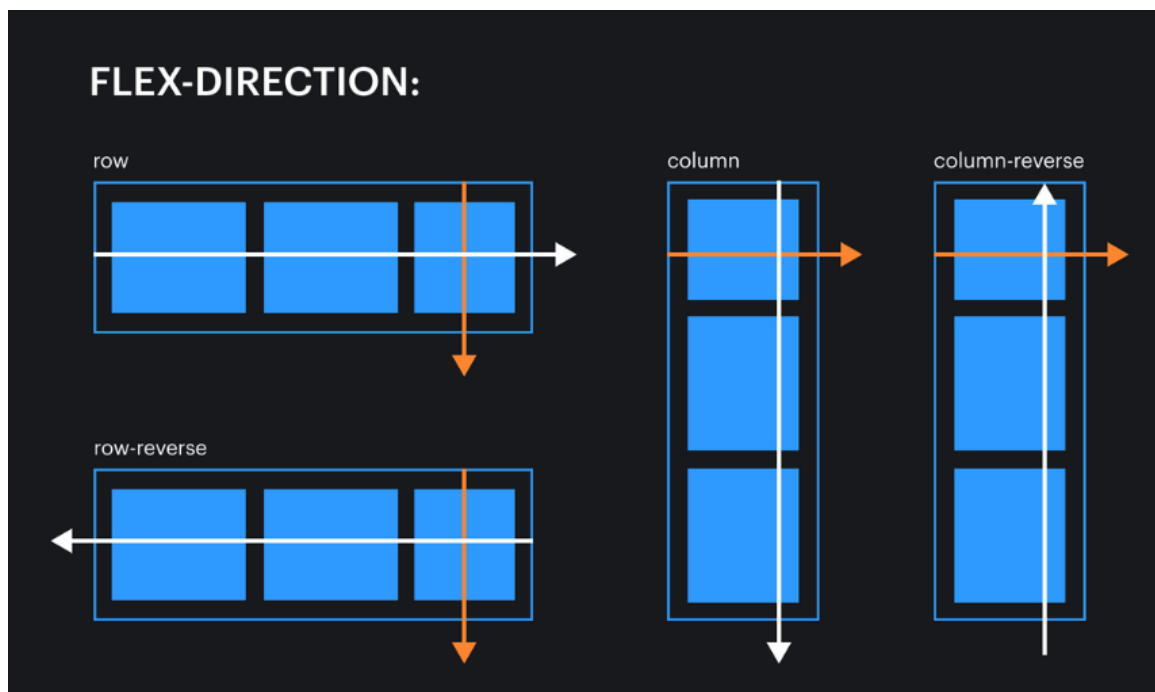


Рисунок 10.3.1 - Свойство flex-direction

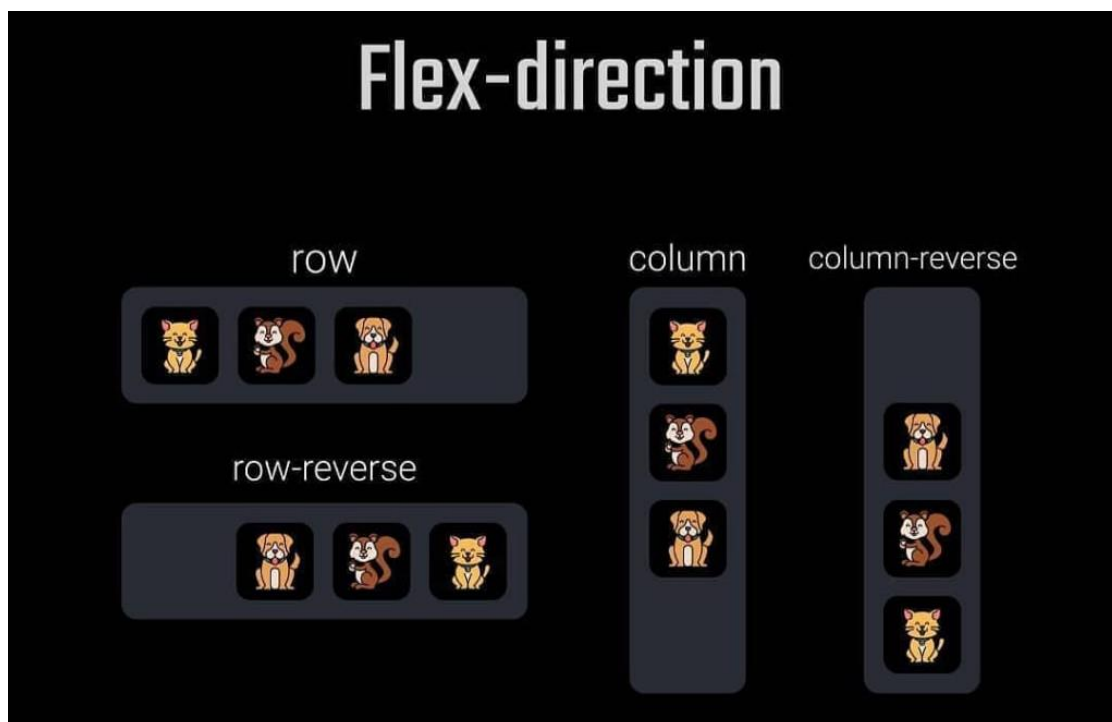


Рисунок 10.3.2 - Свойство flex-direction

### 10.2.3 Свойство `flex-wrap`

Свойство `flex-wrap` определяет, является ли flex-контейнер однострочным или многострочным, а также направление поперечной оси, которое определяет в каком направлении новые строки располагаются друг над другом.

В таблице 10.3.2 обозначены значения свойства `flex-wrap`.

Таблица 10.3.2 - Значения свойства `flex-wrap`

Значение свойства <code>flex-wrap</code>	Описание значения
<code>nowrap</code>	Значение по умолчанию. Flex-элементы помещаются (или пытаются поместиться) в один ряд и не переносятся в новый ряд, даже если не помещаются в размеры родителя.
<code>wrap</code>	Flex-элементы будут иметь возможность переноситься в новый ряд, если не помещаются в одну линию в рамках родителя.
<code>wrap-reverse</code>	Flex-элементы будут располагаться снизу вверх, заполнив собой сперва нижний ряд, а далее те элементы, которые не поместились, перенесутся в ряд выше.

Представление в коде (листинг 10.3.3) и визуально (рисунки 10.3.3, 10.3.4):

Листинг 10.3.3 – Свойство `flex-wrap`

```
.container {  
  display: flex;  
  flex-wrap: nowrap;  
}
```

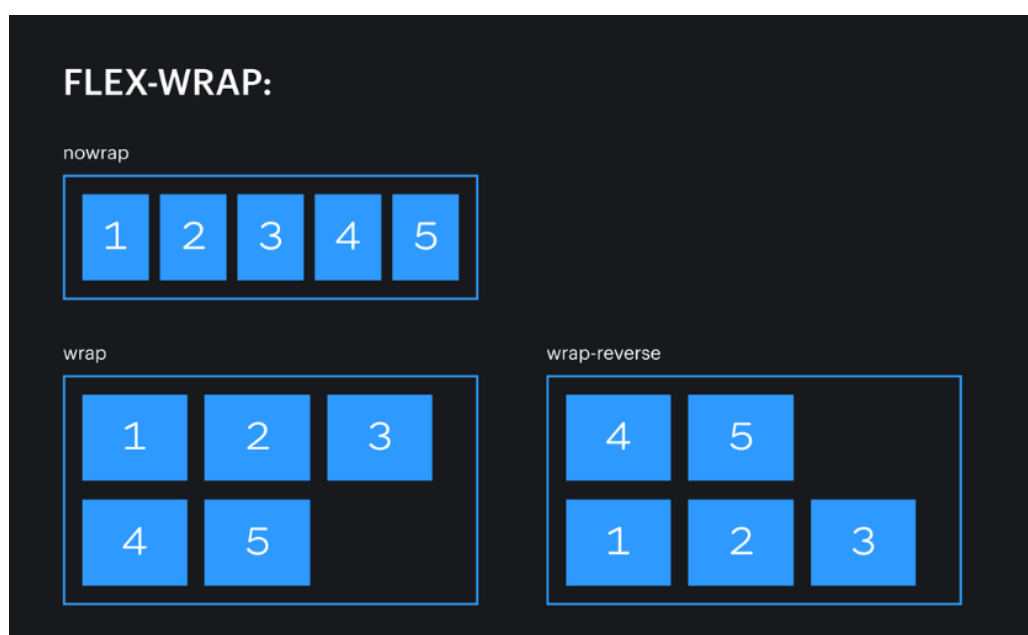


Рисунок 10.3.3 - Свойство `flex-wrap`



Рисунок 10.3.4 - Свойство flex-wrap

#### 10.2.4 Свойство flex-flow

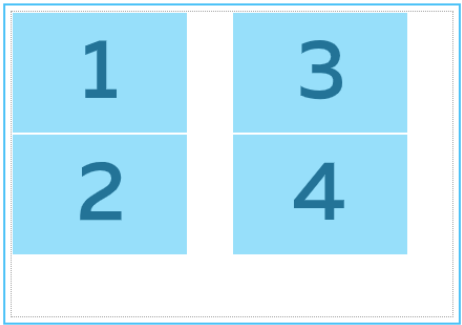
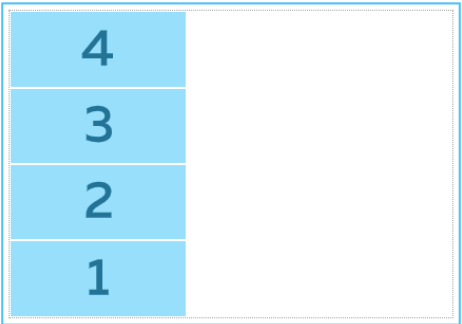
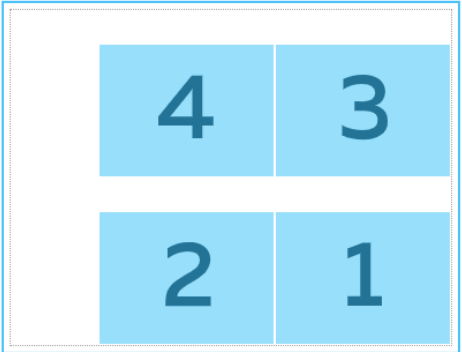
Свойство `flex-flow` — это сокращение для свойств `flex-direction` и `flex-wrap`, которые вместе определяют главную и поперечную оси flex-контейнера.

В таблице 10.3.3 приведены примеры визуальных представлений нескольких значений свойства `flex-flow`.

Таблица 10.3.3 - Значения свойства flex-flow

Значение свойства <code>flex-flow</code>	Представление значения
<code>row nowrap</code>	<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> </div>

Продолжение таблицы 10.3.3

column wrap	
column-reverse nowrap	
row-reverse wrap-reverse	

Представление свойства `flex-flow` в коде (листинг 10.3.4):

Листинг 10.3.4 – Свойство `flex-flow`

```
/* Пример 1 */
.container {
  display: flex;
  flex-flow: column wrap;
}

/* Пример 2 */
.container {
  display: flex;
  flex-flow: row nowrap;
}
```



### 10.2.5 Свойство `justify-content`

Свойство `justify-content` позволяет выравнивать flex-элементы внутри flex-контейнера по основной оси.

Представление свойства `justify-content` в коде (листинг 10.3.5):

Листинг 10.3.5 – Свойство `justify-content`

```
.container {  
  display: flex;  
  justify-content: space-between;  
}
```

В таблице 10.3.4 описаны возможные значения свойства `justify-content`.

Таблица 10.3.4 - Значения свойства `justify-content`

Значение свойства <code>justify-content</code>	Описание значения
<code>flex-start</code>	Значение по умолчанию. Элементы прижимаются к краю, от которого начинается основная ось.
<code>flex-end</code>	Элементы прижимаются к краю, у которого основная ось заканчивается.
<code>start</code>	Элементы прижимаются к тому краю, откуда начинается чтение на том языке, на котором отображается сайт. <i>Например, для русского языка элементы прижмутся к левому краю при горизонтальной основной оси, а для арабского языка — к правому краю.</i>
<code>end</code>	Элементы прижимаются к краю, противоположному началу направления чтения на том языке, на котором отображается сайт. <i>Например, при горизонтальной основной оси на русском языке элементы прижмутся к правому краю.</i>
<code>left</code>	Элементы прижмутся к левому краю родителя. В случае, если указано свойство <code>flex-direction: column</code> , значение срабатывает как <code>start</code> .
<code>right</code>	Элементы прижмутся к правому краю родителя. В случае, если указано свойство <code>flex-direction: column</code> , значение срабатывает как <code>end</code> .
<code>center</code>	Элементы выстраиваются по центру родителя.
<code>space-between</code>	Крайние элементы прижимаются к краям родителя, а оставшиеся выстраиваются внутри контейнера равномерно, так, чтобы между ними были одинаковые отступы.
<code>space-around</code>	Свободное пространство делится поровну между элементами и по половине от этой доли размещается по бокам от каждого элемента.

	Таким образом, между соседними элементами будет равное расстояние, а снаружи крайних элементов — по половине этого расстояния.
<code>space-evenly</code>	Свободное место будет распределено так, чтобы расстояние между любыми двумя элементами было одинаковым и расстояние от крайних элементов до края было таким же.

Визуальное представление основных значений свойства `justify-content` представлено на рисунке 10.3.5:

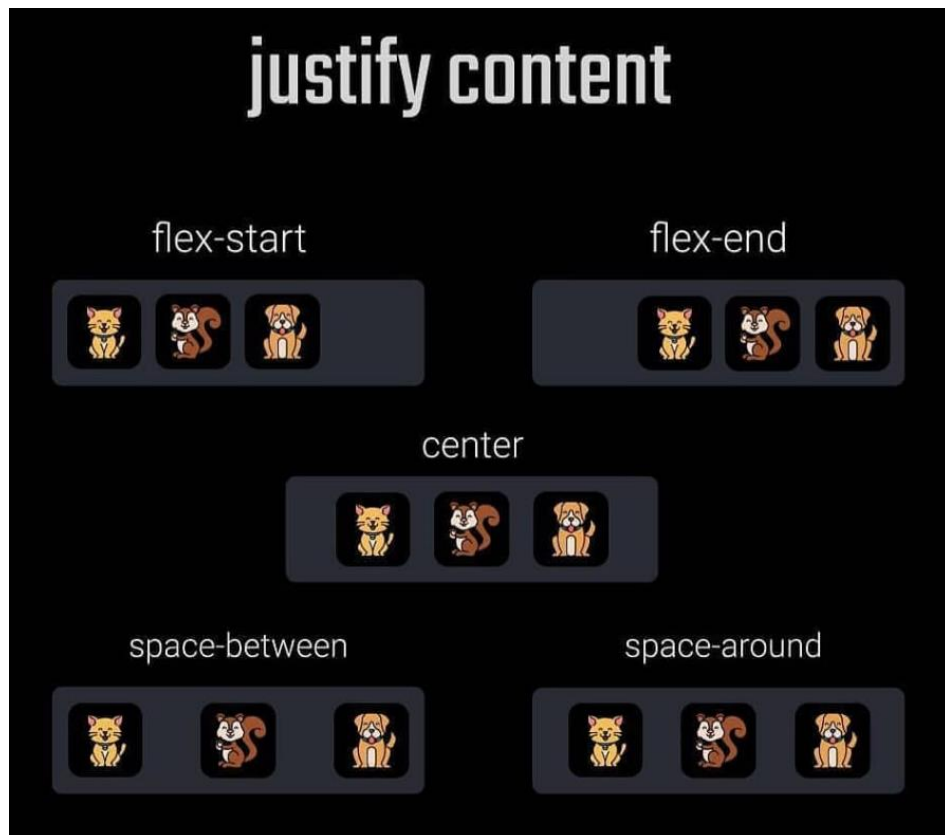


Рисунок 10.3.5 - Свойство `justify-content`

### 10.2.6 Свойство `align-items`

Свойство `align-items` позволяет выравнивать элементы внутри flex-контейнера по поперечной оси.

Представление свойства `align-items` в коде (листинг 10.3.6):

Листинг 10.3.6 – Свойство `align-items`

```
.container {
  display: flex;
  align-items: center;
}
```

В таблице 10.3.5 описаны возможные значения свойства `align-items`.

Таблица 10.3.5 - Значения свойства `align-items`

Значение свойства <code>align-items</code>	Описание значения
<code>stretch</code>	Значение по умолчанию. Элементы растягиваются вдоль поперечной оси так, чтобы заполнить всего родителя. <i>Это очень удобно, если вы делаете макет из двух колонок.</i>
<code>flex-start</code> / <code>start</code>	Элементы выстраиваются у начала поперечной оси. Значение <code>start</code> зависит от направления чтения выбранного языка.
<code>flex-end</code> / <code>end</code>	Элементы выстраиваются у конца поперечной оси. Значение <code>end</code> зависит от направления чтения выбранного языка.
<code>center</code>	Элементы выстраиваются по центру поперечной оси.
<code>baseline</code>	Элементы выравниваются по базовой линии текста. «Базовая линия» — <code>baseline</code> — воображаемая линия, проходящая по нижнему краю знаков шрифта (без учёта выносных элементов).

Визуальное представление значений свойства `align-items` представлено на рисунке 10.3.6:



Рисунок 10.3.6 - Свойство `align-items`

### 10.2.7 Свойство align-content

Свойство `align-content` распределяет свободное пространство по поперечной оси между рядами flex-элементов.

Представление свойства `align-content` в коде (листинг 10.3.7):

Листинг 10.3.7 – Свойство align-content

```
.container {  
  display: flex;  
  align-content: center;  
}
```

В таблице 10.3.6 описаны возможные значения свойства `align-content`.

Таблица 10.3.6 - Значения свойства align-content

Значение свойства align-content	Описание значения
<code>stretch</code>	Значение по умолчанию. Ряды растягиваются одинаково, так, чтобы занять всё доступное пространство родителя.
<code>flex-start</code> / <code>start</code>	Все ряды располагаются у начала поперечной оси. <i>Значение <code>start</code> зависит от направления чтения выбранного языка.</i>
<code>flex-end</code> / <code>end</code>	Все ряды располагаются у конца поперечной оси. <i>Значение <code>end</code> зависит от направления чтения выбранного языка.</i>
<code>center</code>	Ряды выравниваются по центру родителя.
<code>space-between</code>	Первый ряд прижимается к началу поперечной оси, последний — к концу поперечной оси, а остальные располагаются так, чтобы свободное пространство было поделено на отступы между ними равномерно.
<code>space-around</code>	Отступы у каждого ряда равнозначны отступам у любого другого ряда.
<code>space-evenly</code>	Отступы между рядами и от краёв родителя одинаковые.

Визуальное представление значений свойства `align-content` представлено на рисунке 10.3.7:

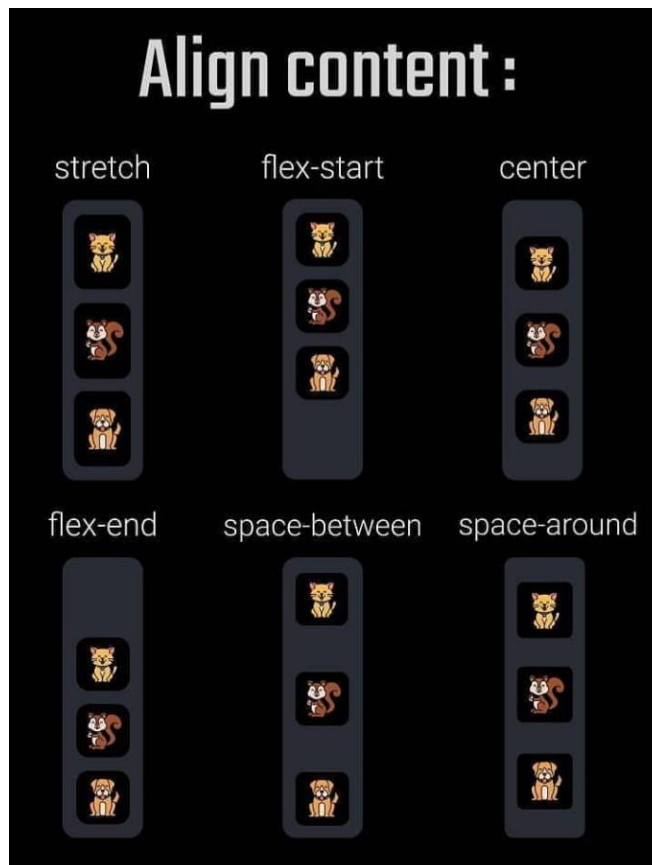


Рисунок 10.3.7 - Свойство align-content

### 10.2.8 Свойство gap

Свойство `gap` задает отступы между строками и столбцами. Является краткой записью свойств `row-gap` и `column-gap`.

Может иметь одно или два значения. Если указано только одно, то `column-gap` автоматически равен `row-gap`. Если указаны два значения, то первое будет задавать отступы между рядами (`row-gap`), а второе — между колонками (`column-gap`).

Значения можно указывать в любых доступных единицах измерения, включая проценты. Допускается использование функции `calc()`.

Представление свойства `gap` в коде (листинг 10.3.8):

Листинг 10.3.8 – Свойство gap

```
.container {  
  display: flex;  
  gap: 30px calc(10rem - 10px);  
}
```

## 10.3 Свойства flex-элементов

### 10.3.1 Свойство order

При помощи свойства `order` можно менять порядок отображения flex-элементов внутри flex-контейнера.

По умолчанию элементы отображаются в том порядке, в котором они расположены в разметке, а значение свойства `order` равно 0.

Значение задаётся в виде целого отрицательного или положительного числа. Элементы встают по возрастающей.

Представление свойства `order` в коде (листинг 10.4.1):

Листинг 10.4.1 – Свойство `order`

```
.container {  
  display: flex;  
}  
  
.item {  
  order: 3;  
}
```

Визуальное представление значений свойства `order` представлено на рисунке 10.4.1:

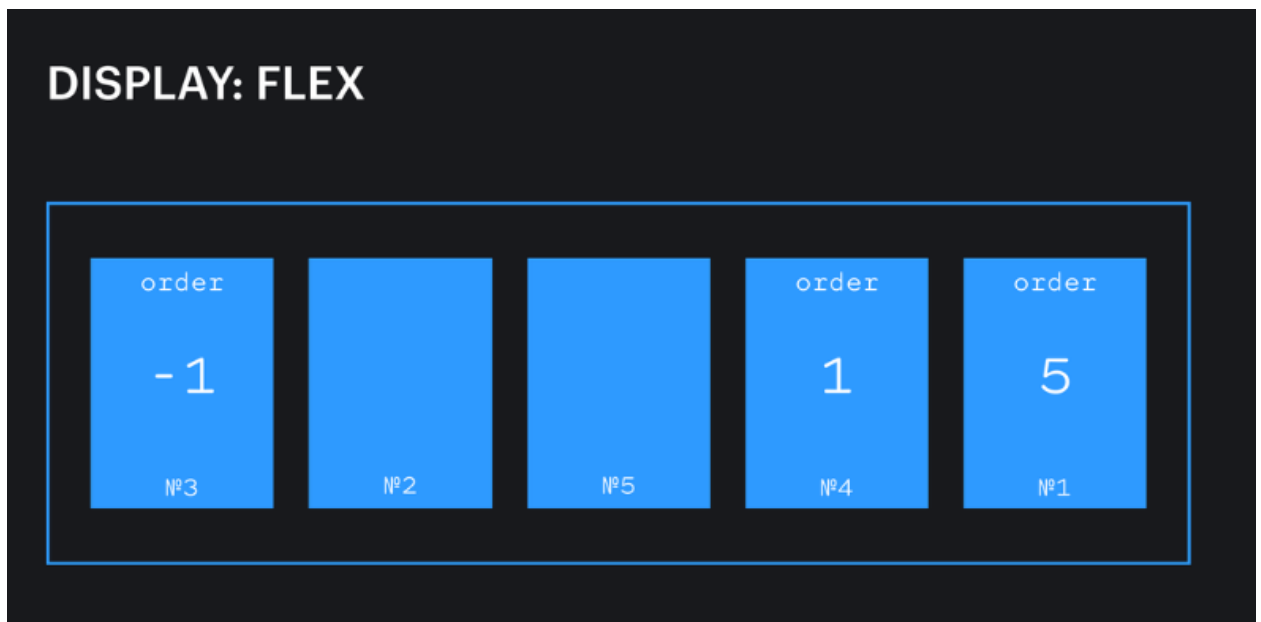


Рисунок 10.4.1 - Свойство `order`

### 10.3.2 Свойство `flex-grow`

Свойство `flex-grow` указывает, может ли вырастать flex-элемент при наличии свободного места, и насколько. По умолчанию значение равно `0`. Значением может быть **любое положительное целое число** (включая `0`).

Если у всех flex-элементов будет прописано `flex-grow: 1`, то свободное пространство в контейнере будет равномерно распределено между всеми.

Если при этом одному из элементов будет задано `flex-grow: 2`, то он займет в два раза больше свободного места, чем его соседи.

Представление свойства `flex-grow` в коде (листинг 10.4.2):

Листинг 10.4.2 – Свойство `flex-grow`

```
.container {  
  display: flex;  
}  
  
.item {  
  flex-grow: 1;  
}
```

Визуальное представление значений свойства `flex-grow` представлено на рисунке 10.4.2:



Рисунок 10.4.2 - Свойство `flex-grow`

### 10.3.3 Свойство `flex-shrink`

Свойство `flex-shrink` полностью противоположно свойству `flex-grow`. Если в контейнере не хватает места для расположения всех элементов без изменения размеров, то свойство `flex-shrink` указывает, в каких пропорциях элемент будет уменьшаться.

Чем больше значение у этого свойства, тем быстрее элемент будет сжиматься по сравнению с соседями, имеющими меньшее значение.

Значение по умолчанию — 1. Значением может быть любое целое положительное число (включая 0).

Два предыдущих свойства работают с пропорциональным разделением пространства, не с конкретными размерами.

Представление свойства `flex-shrink` в коде (листинг 10.4.3):

Листинг 10.4.3 – Свойство `flex-shrink`

```
.container {  
  display: flex;  
}  
  
.item {  
  flex-shrink: 3;  
}
```

Визуальное представление значений свойства `flex-shrink` представлено на рисунке 10.4.3:



Рисунок 10.4.3 - Свойство `flex-shrink`

#### 10.3.4 Свойство `flex-basis`

Свойство `flex-basis` указывает на размер элемента до того, как свободное место будет распределено.

Значением может быть размер в любых относительных или абсолютных единицах: `20rem`, `5vw`, `250px`. А также можно использовать ключевое слово `auto` (значение по умолчанию). В этом случае при расчёте размера элемента будут приниматься во внимание



значения свойств `width`, `max-width`, `min-width` или аналогичные свойства высоты, в зависимости от того, в каком направлении идёт основная ось.

Если никакие размеры не заданы, а свойству `flex-basis` установлено значение `auto`, то элемент занимает столько пространства, сколько нужно для отображения контента.

Представление свойства `flex-basis` в коде (листинг 10.4.4):

Листинг 10.4.4 – Свойство `flex-basis`

```
.container {
  display: flex;
  flex-wrap: wrap;
  align-content: center;
  height: 100%;
}
.item {
  flex-basis: 30%;
}
```

Пример выполнения вышепредставленного кода (рисунок 10.4.4):

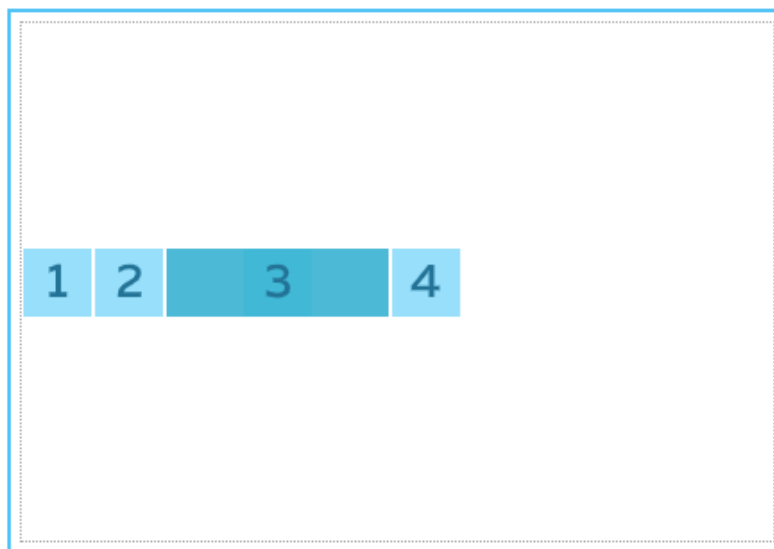


Рисунок 10.4.4 – Свойство `flex-basis`

### 10.3.5 Свойство `flex`

С помощью свойства `flex` можно указать значение **трёх свойств одновременно**: `flex-grow`, `flex-shrink` и `flex-basis`. Первое значение является обязательным, остальные опциональны.

Значение по умолчанию: `0 1 auto`, что расшифровывается как `flex-grow: 0`, `flex-shrink: 1`, `flex-basis: auto`.

Примеры возможных значений свойства `flex` (листинг 10.4.5):

Листинг 10.4.5 – Свойство `flex` с разными значениями

```
.item {  
  
  /* 0 0 auto */  
  flex: none;  
  
  /* Одно значение, число без единиц: flex-grow */  
  flex: 2;  
  
  /* Одно значение, ширина/высота: flex-basis */  
  flex: 10em;  
  flex: 30px;  
  flex: auto;  
  flex: content;  
  
  /* Два значения: flex-grow | flex-basis */  
  flex: 1 30px;  
  
  /* Два значения: flex-grow | flex-shrink */  
  flex: 2 2;  
  
  /* Три значения: flex-grow | flex-shrink | flex-basis */  
  flex: 2 2 10%;  
  
  /* Глобальные значения */  
  flex: inherit;  
  flex: initial;  
  flex: unset;  
  
}
```

### 10.3.6 Свойство `align-self`

При помощи свойства `align-self` можно выровнять один или несколько элементов иначе, чем задано у родительского элемента. Например, в коде ниже у родителя задано выравнивание вложенных элементов по верхнему краю родителя. А для элемента с классом `.item` мы задаём выравнивание по нижнему краю.

Представление свойства `align-self` в коде (листинг 10.4.6):

Листинг 10.4.6 – Свойство `align-self`

```
.container {  
  display: flex;  
  align-items: flex-start;  
}  
  
.item {  
  align-self: flex-end;  
}
```

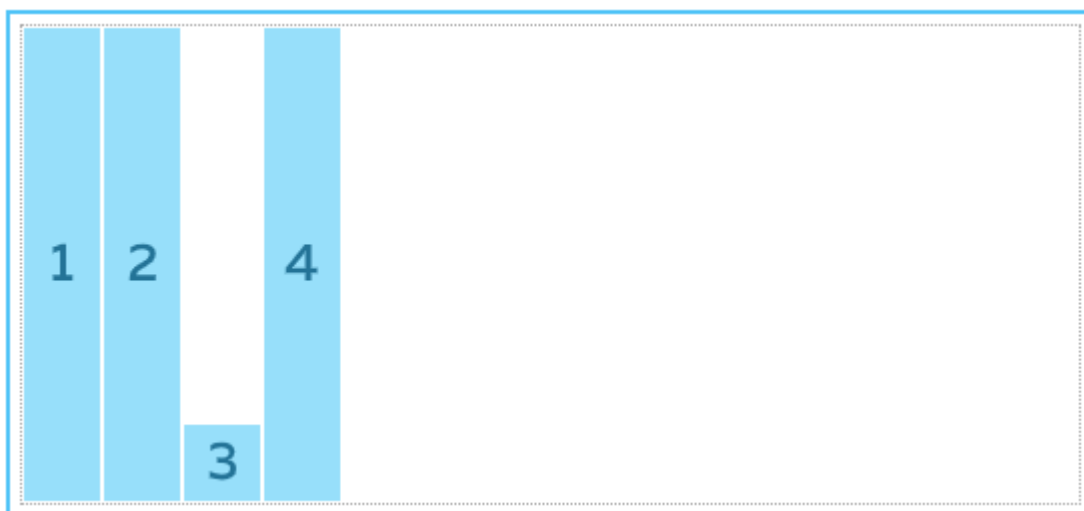


Рисунок 10.4.5 – Свойство align-self (flex-end)

Примеры сайтов, ориентированных с помощью медиа-запросов и flexbox, можно посмотреть на сайте Mediaqueri.es – URL: <https://mediaqueri.es/>.

#### Дополнительная информация

1. Спецификация по flexbox – URL: <https://www.w3.org/TR/css-flexbox-1/#flex-containers> (англ. яз.)
2. Игра по flexbox - Game: Flexbox Froggy - URL: <https://flexboxfroggy.com/#ru> (русс. яз.)
3. Игра по flexbox - Game: Flexbox Defense - URL: <http://www.flexboxdefense.com/> (англ. яз.)
4. Игра по flexbox - Game: Flexbox Ducky - URL: <https://courses.cs.washington.edu/courses/cse154/flexboxducky/> (англ. яз.)
5. A Complete Guide to Flexbox – URL: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/> (англ. яз.)
6. Flexbox - удобная шпаргалка по флексам – URL: <https://codepen.io/enxaneta/pen/adLPwv> (англ. яз.)

## Практическое задание

1. Создать с помощью **Flexbox** адаптивные макеты **своих сайтов** под устройства как показано на рисунках-примерах (макеты можно модифицировать под свою семантическую структуру сайта):

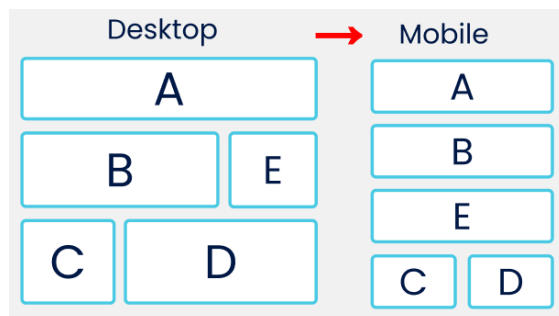


Рисунок 1

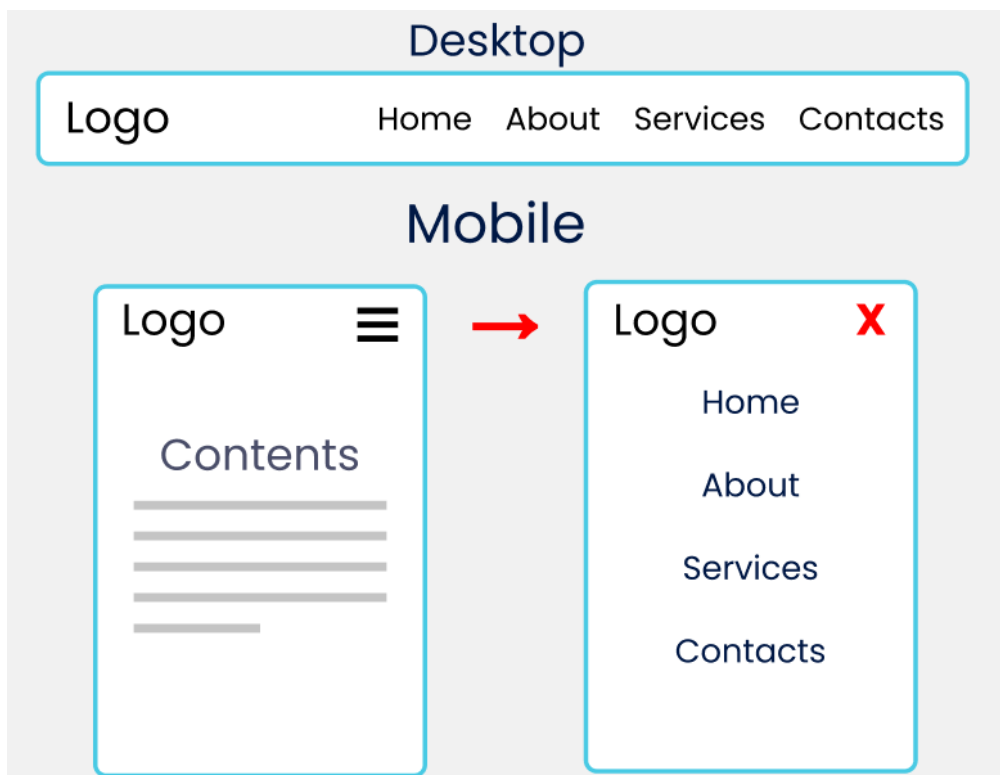


Рисунок 2

### Примечания для макета на рисунке 2:

1. Используйте свойства `visibility` и `opacity`.

Свойство `visibility` показывает и прячет элемент от глаз пользователя практически так же, как это делает `opacity`. И в том, и в другом случае элемент не виден, но механизм работы этих свойств разный. Если при помощи `opacity` можно

гибко изменять прозрачность элемента и делать его, например, видимым на 33% (0 – полная прозрачность, 1 – полная непрозрачность), то свойство `visibility` имеет только два состояния: `visible` (видимый) и `hidden` (элемент не виден на странице, но занимает отведённое ему место и влияет на поток документа).

2. Используйте комбинаторы наследования для меню-бургера.
3. Используйте псевдоэлементы `::before` и `::after` для меню-бургера.
4. Используйте псевдокласс `:checked` для меню-бургера.
5. Используйте свойство `transform` со значениями `rotate(0/45deg/90deg)` (т.е. вращение в градусах) для работы с меню-бургером.

2. Создать адаптивную галерею изображений по тематике своего сайта с применением медиа-запросов и **Flexbox**. Примеры галерей: <https://unsplash.com/>, <https://duotone.shapefactory.co/>, <https://www.pinterest.ru/>.