CSS Flexbox позволяет создать адаптивный дизайн намного проще, чем с использованием Float и позиционирования.

Flexbox можно использовать как для CSS разметки целой страницы, так и её отдельных блоков.

### Поддержка CSS Flexbox браузерами

CSS Flexbox поддерживается всеми используемые на сегодняшний момент современными браузерами (с использованием префиксов: IE10+, Edge12+, Firefox 2+, Chrome 4+, Safari 3.1+, Opera 12.1+, iOS Safari 3.2, Opera mini, Android 2.1+, Blackberry 7+).

## Основы CSS Flexbox

Создание CSS разметки с помощью Flexbox начинается с установки необходимому HTML элементу CSS-свойства display со значением flex или flex-inline.

После этого данный элемент становится flex-контейнером, а все его непосредственные дочерние элементы – flex-элементами. При этом когда мы говорим о flexbox то подразумеваем под этим только элемент с display:flex или display:flex-inline и все элементы непосредственно расположенные в нём. Таким образом в CSS Flexbox имеется всего два типа элементов: flex-контейнер и flex-элемент.

<style>

.flex-container {

display: flex; /\* flex || inline-flex \*/

}

<style>

<!-- flex-контейнер -->

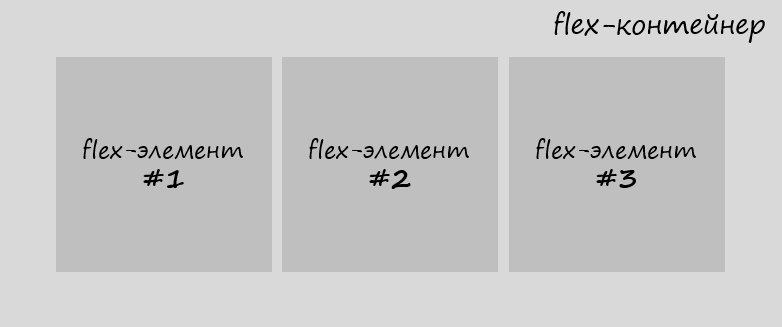
<div class="flex-container">

<div>flex-элемент #1</div>

<div>flex-элемент #2</div>

<div>flex-элемент #3</div>

</div>

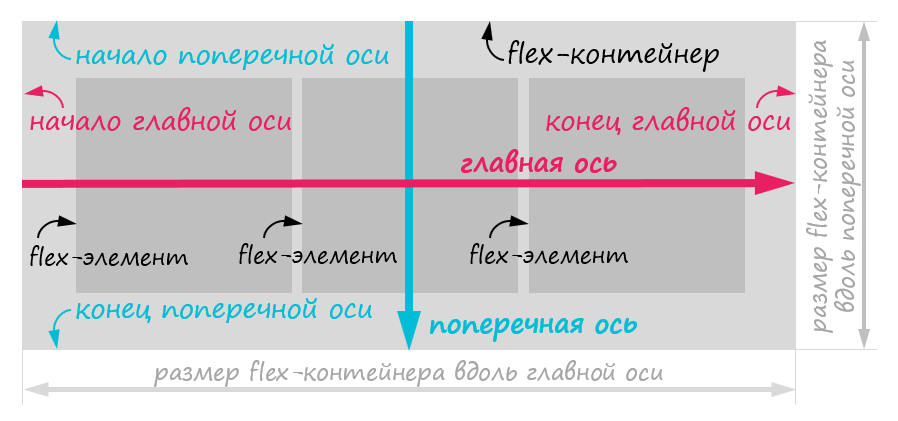


По умолчанию flex-элементы во flex-контейнере занимают всю его высоту.

Значение flex или flex-inline определяет то, как flex-контейнер будет представлен на странице. Если его необходимо отобразить в виде блока, то используйте значение flex. Если элемент необходимо представить как строку, то используйте значение flex-inline. В этом случае он будет занимать столько места странице, сколько необходимо для отображения его элементов.

### Устройство flex-контейнера. Направление осей

На рисунке представлена схема устройства flex-контейнера:



Направление расположение flex-элементы в flex-контейнере определяется посредством осей.

В CSS Flexbox имеются две оси. Первая ось называется главной (по умолчанию она направлена слева направо). Вторая - поперечная (по умолчанию направлена сверху вниз), она всегда перпендикулярно главной. Главная ось задаёт основное направление flex-элементов во flex-контейнере, а поперечная ось определяет их направление при переносе на новую линию.

По умолчанию элементы во flex-контейнере располагаются вдоль направления главной оси (т.е. слева направо) на одной линии.

Направление главной оси можно изменить, осуществляется это с помощью CSS-свойства flex-direction.

flex-direction: row;

/\* row (слева направо) - по умолчанию

row-reverse (справа налево)

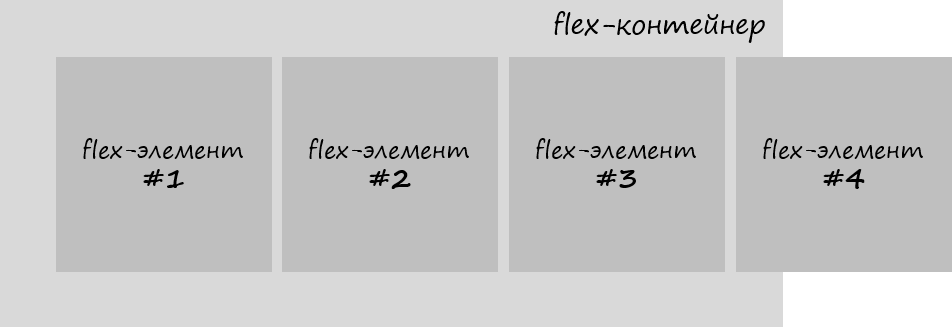
column (сверху вниз)

column-reverse (снизу вверх) \*/

С помощью этого свойства можно сделать так, чтобы flex-элементы располагались не рядами (rows), а колонками (columns). Осуществляется это с помощью значения column или column-reverse.



По умолчанию flex-элементы не переносятся на новую линию, даже когда им не хватает места в текущей линии. Они просто выходят за её пределы.



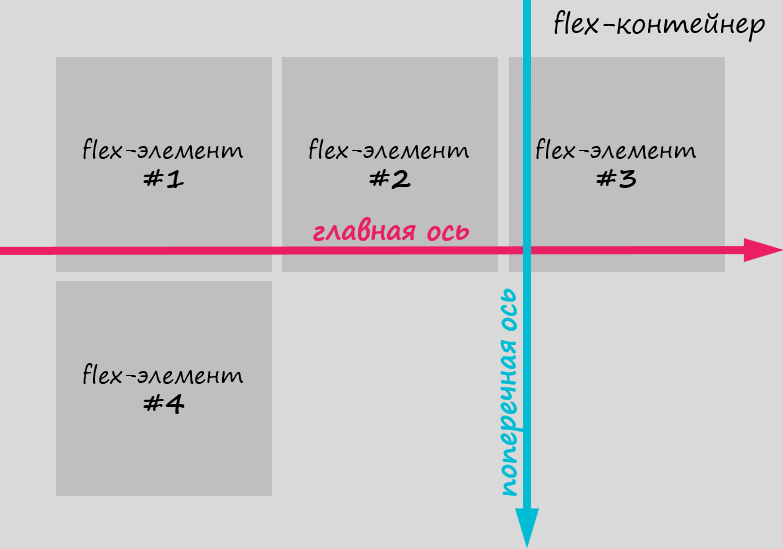
Но это можно изменить. Разрешить перенос flex-элементов на новые линии осуществляется с помощью установки flex-контейнеру CSS свойства flex-wrap со значением wrap или wrap-reverse.

flex-wrap: wrap;

/\* nowrap (только на одной линии - по умолчанию)

wrap (разрешить перенос flex-элементов на новые линии)

wrap-reverse (осуществлять перенос flex-элементов в обратном порядке) \*/



Значения wrap и wrap-reverse CSS-свойства flex-wrap определяют направление поперечной оси.

Свойства flex-direction и flex-wrap можно указать с помощью универсального CSS свойства flex-flow:

flex-flow: row nowrap; /\* 1 значение - flex-direction, 2 значение - flex-wrap \*/

## Выравнивание flex-элементов

Во Flexbox выравнивание элементов внутри контейнера осуществляется по двум направлениям (осям).

### Выравнивание flex-элементов по направлению главной оси

Выравнивание элементов вдоль основной оси осуществляется с помощью CSS свойства justify-content:

justify-content: flex-start;

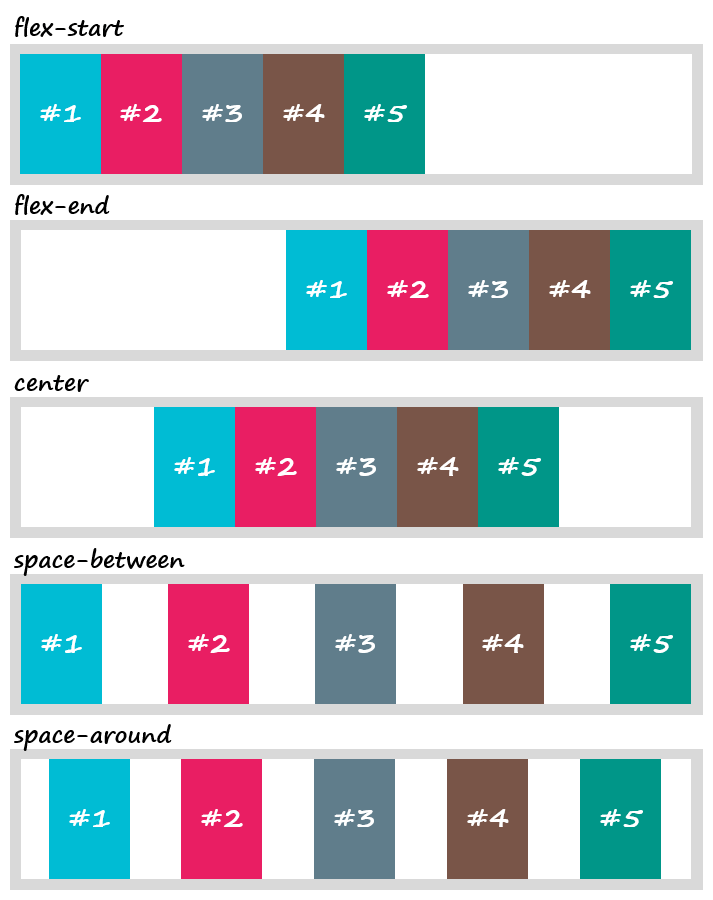
/\* flex-start (flex-элементы выравниваются относительно начала оси) – по умолчанию

flex-end (flex-элементы выравниваются относительно конца оси)

center (по центру flex-контейнера)

space-between (равномерно, т.е. с одинаковым расстоянием между flex-элементами)

space-around (равномерно, но с добавлением половины пространства перед первым flex-элементом и после последнего) \*/



### Выравнивание flex-элементов вдоль поперечной оси

Выравнивание flex-элементов во flex-контейнере по направлению поперечной оси осуществляется с помощью CSS-свойства align-items:

align-items: stretch;

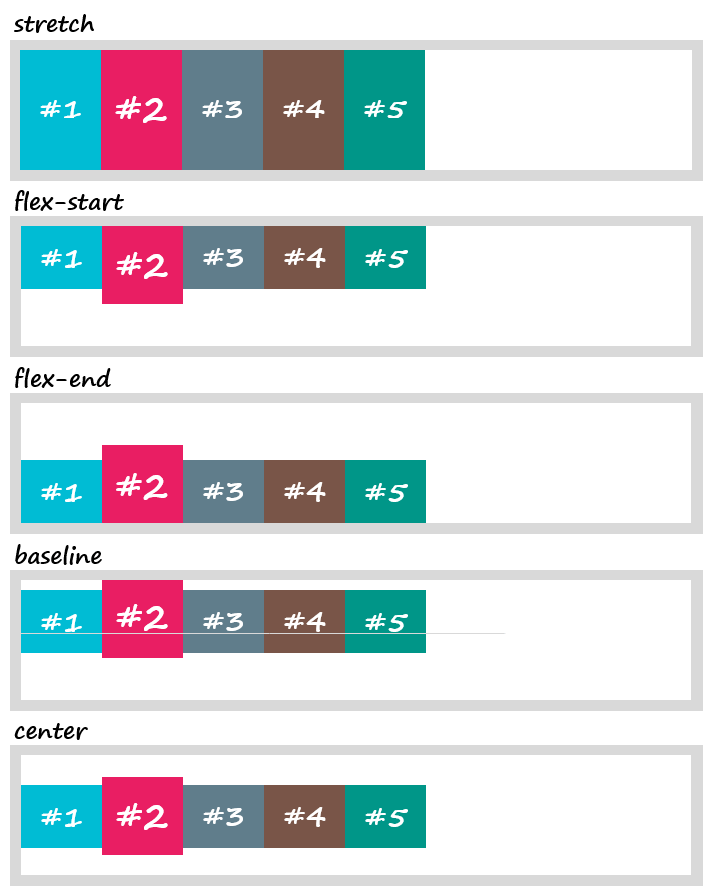
/\* stretch (растягиваются по всей длине линии вдоль направления поперечной оси) – по умолчанию

flex-start (располагаются относительно начала поперечной оси)

flex-end (относительно конца поперечной оси)

baseline (относительно базовой линии)

center (по центру) \*/



### Выравнивание линий flex-контейнера

CSS Flexbox позволяет выравнивать не только сами flex-элементы, но и линии на которых они расположены.

align-content: stretch

/\* stretch (растягиваются по всей длине поперечной оси) – по умолчанию

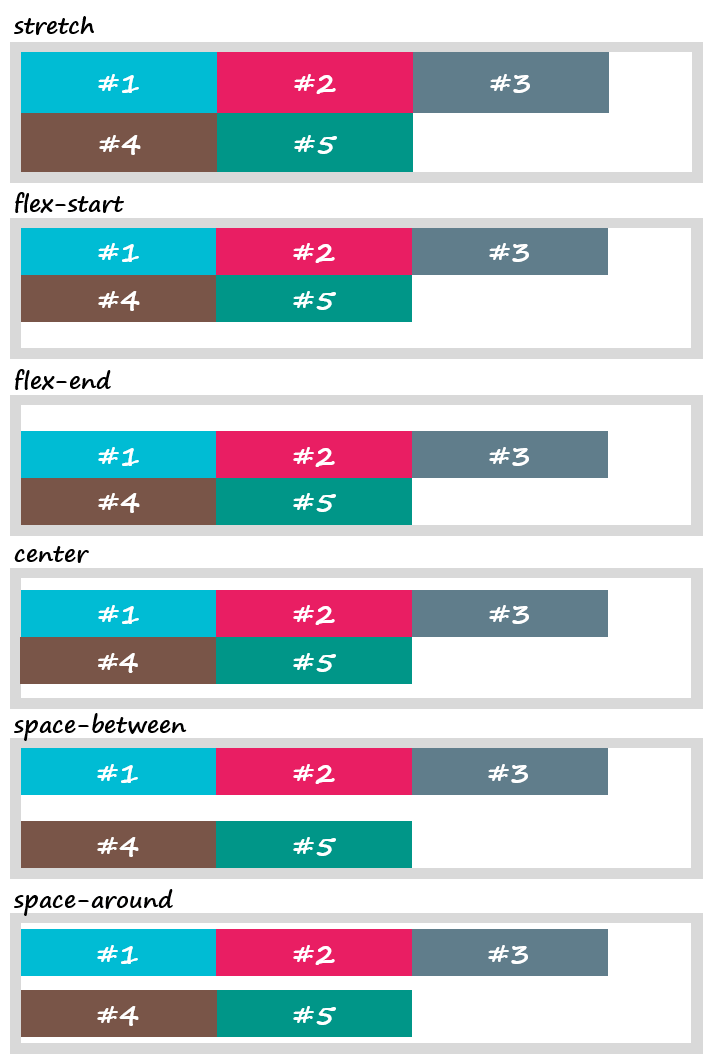
flex-start (относительно начала поперечной оси)

flex-end (относительно конца поперечной оси)

center (по центру)

space-between (равномерно, т.е. с одинаковым расстоянием между линиями)

space-around (равномерно, но с добавлением половины пространства перед первой линией и после последней) \*/



Свойство align-content имеет смысл использовать только тогда, когда flex-элементы во flex-контейнере располагаются на нескольких линиях. Чтобы это произошло, необходимо, во-первых, чтобы ширина всех flex-элементов была больше ширины flex-контейнера, а во-вторых flex-контейнер должен иметь в качестве CSS-свойства flex-wrap значение wrap или wrap-reverse.

### CSS-свойство align-self

Свойство align-self в отличие от предыдущих (justify-content, align-items и align-content) предназначено для flex-элементов. Оно позволяет изменить **выравнивание flex-элемента** вдоль направления поперечной оси. Свойство align-self может принимать такие же значения как align-items.

align-items: stretch; /\* auto (по умолчанию) || stretch || flex-start || flex-end || baseline || center \*/

Пример:

<div class="flex-container">

<div class="flex-container\_element-1">

1

</div>

<div class="flex-container\_element-2">

2

</div>

<div class="flex-container\_element-3">

3

</div>

<div class="flex-container\_element-4">

4

</div>

</div>

CSS:

.flex-container {

display: flex;

width: 300px;

height: 150px;

align-items: center;

padding: 10px;

background-color: #efefef;

}

.flex-container\_element-1,

.flex-container\_element-2,

.flex-container\_element-3,

.flex-container\_element-4 {

flex-basis: 70px;

text-align: center;

padding: 15px;

font-size: 30px;

}

.flex-container\_element-1 {

align-self: flex-start;

background: #fe4;

}

.flex-container\_element-2 {

align-self: flex-end;

background: pink;

}

.flex-container\_element-3 {

align-self: stretch;

background: lime;

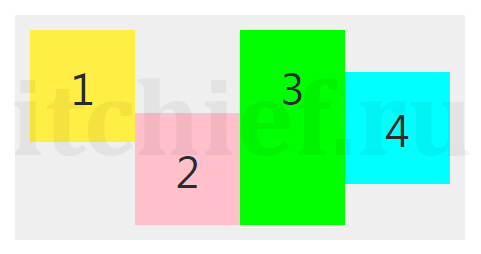
}

.flex-container\_element-4 {

align-self: auto;

background: cyan;

}



## Изменение порядка следования flex-элементов

По умолчанию flex-элементы отображаются во flex-контейнере в том порядке, в котором они расположены в HTML коде. Для изменения порядка следования одних flex-элементов относительно других в CSS Flexbox можно использовать свойство order. Данное CSS свойство выстраивает flex-элементы во flex-контейнере в порядке возрастания их номеров.

order: 0;

/\*

0 (по умолчанию)

целое положительное или отрицательное число

\*/

Например:

<div class="flex-container">

<div class="flex-container\_element-1">...</div>

<div class="flex-container\_element-2">...</div>

<div class="flex-container\_element-3">...</div>

<div class="flex-container\_element-4">...</div>

</div>

CSS:

.flex-container {

display: flex;

}

/\* переместим 2 flex-элемент в конец \*/

.flex-container\_element-2 {

order: 2;

}

/\* передвинем 3 элемент до 2 \*/

.flex-container\_element-3 {

order: 1;

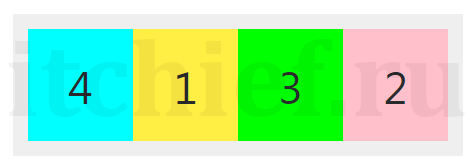
}

/\* расположим 4 flex-элемент до 1 \*/

.flex-container\_element-4 {

order: -1;

}



## Управление шириной flex-элемента

Во Flexbox есть несколько CSS свойств, определяющих то, какая ширина может быть у flex-элемента.

### CSS-свойство flex-basis

Данное свойство предназначено для **установления начальной ширины flex-элементу**. Задавать значение ширины можно посредством различных единиц измерения, таких как px, %, em и др. По умолчанию данное свойство имеет значение auto (в этом случае ширина элемента будет рассчитываться автоматически на основании его содержимого).

**Конечная ширина flex-элемента будет определяться** в зависимости от значений CSS-свойств flex-grow и flex-shrink, которые установлены не только для этого элемента, но и для других flex-элементов этого flex-контейнера.

### CSS-свойство flex-grow

Это свойство определяет, может ли **начальная ширина flex-элемента увеличиваться (расти)**. Увеличение ширины flex-элемента осуществляется за счёт **свободного пространства линии**. В качестве значения CSS-свойства flex-grow указывается **целое число**. Именно это значение и определяет (если оно больше или равно 1) какую часть свободного пространства flex-элемент заберёт себе.

Например:

<div class="flex-container">

<div class="flex-container\_element-1">...</div>

<div class="flex-container\_element-2">...</div>

</div>

CSS:

.flex-container {

display: flex;

width: 600px;

}

.flex-container\_element-1 {

flex-basis: 40%;

flex-grow: 1;

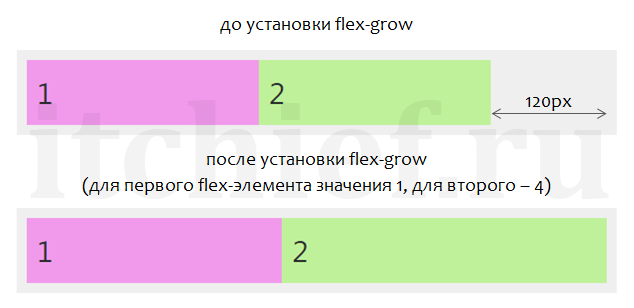
}

.flex-container\_element-2 {

flex-basis: 40%;

flex-grow: 4;

}



В этом примере, если flex-элементы расположены на одной линии и в ней есть свободное пространство (600×(1-0,8)=120px):

* к ширине элемента .flex-container\_element-1 добавится 1/5 часть этого пространства (120×1/5=24px);
* к ширине элемента .flex-container\_element-2 добавится 4/5 части этого пространства (120×4/5=96px).

Другими словами, CSS свойство flex-grow позволяет не просто указать, что ширина flex-элемента может вырасти, но и задать, насколько эта величина может вырасти по отношению к другим элементам.

По умолчанию CSS свойство flex-grow имеет значение 0. Это означает, что flex-элемент не может расти (увеличивать свою ширину).

### CSS-свойство flex-shrink

Данное свойство определяет, может ли ширина flex-элемента уменьшиться. Уменьшение ширины flex-элемента будет осуществляться только в том случае, если **ширины линии будет не достаточно для отображения всех flex-элементов**, расположенных в ней. Необходимая ширина рассчитывается на **основании начальной ширины**, который имеет каждый flex-элемент в ней.

Например:

<div class="flex-container">

<div class="flex-container\_element-1">...</div>

<div class="flex-container\_element-2">...</div>

</div>

CSS:

.flex-container {

display: flex;

width: 500px;

}

.flex-container\_element-1 {

flex-basis: 300px;

flex-shrink: 1;

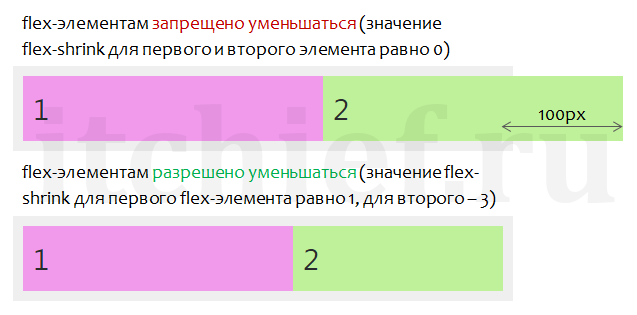
}

.flex-container\_element-2 {

flex-basis: 300px;

flex-shrink: 3;

}



Ширина flex-контейнера 500px. Для отображения flex-элементов необходимо 600px. В итоге не хватает 100px. В этом примере уменьшаться могут 2 flex-элемента (.flex-container\_element-1 и .flex-container\_element-2). Ширина первого flex-элемента .flex-container\_element-1 в данном случае составит 300 – 1/4\*100= 275px. Ширина второго flex-элемента .flex-container\_element-2 в данном случае составит 300 – 3/4\*100= 225px.

Значение по умолчанию:

flex-shrink: 1;

Если вам необходимо **запретить уменьшение ширины flex-элементу**, то в качестве значения свойства flex-shrink необходимо указать число 0.

### CSS-свойство flex

Для удобной установки flex-grow, flex-shrink и flex-basis можно использовать CSS свойство flex.

Значение по умолчанию:

flex: 0 1 auto;

/\*

0 - flex-grow (1 значение)

1 - flex-shrink (2 значение)

auto - flex-basis (3 значение)

\*/

## Пример верстки макета на CSS Flexbox

В этом разделе создадим простой адаптивный макет всей страницы на CSS Flexbox.

Структура макета будет состоять из 3 секций:

* header (для вывода заголовка и основного меню);
* main (для отображения основной части);
* footer (для футера).

Основную часть (main) в свою очередь разделим ещё на 2 раздела (их позиционирование будем осуществлять с помощью CSS Flexbox). На больших экранах (>=992px) эти разделы выстроим горизонтально, а на остальных - вертикально (<992px).

<style>

/\* контейнер \*/

.container {

width: 100%;

max-width: 1200px;

padding-right: 15px;

padding-left: 15px;

margin-left: auto;

margin-right: auto;

}

/\* flex-контейнер \*/

.row {

display: -ms-flexbox;

display: flex;

-ms-flex-wrap: wrap;

flex-wrap: wrap;

margin-right: -15px;

margin-left: -15px;

}

/\* общие настройки для flex-элементов \*/

.col {

position: relative;

width: 100%;

padding-right: 15px;

padding-left: 15px;

}

/\* ширина блоков .col\_\_article и .col\_\_aside по умолчанию \*/

.col\_\_article,

.col\_\_aside {

-ms-flex: 0 0 100%;

flex: 0 0 100%;

max-width: 100%;

}

/\* ширина блоков .col\_\_article и .col\_\_aside для больших экранов \*/

@media (min-width: 992px) {

/\* 2/3 от ширины контейнера \*/

.col\_\_article {

-ms-flex: 0 0 66.666667%;

flex: 0 0 66.666667%;

max-width: 66.666667%;

}

/\* 1/3 от ширины контейнера \*/

.col\_\_aside {

-ms-flex: 0 0 33.333333%;

flex: 0 0 33.333333%;

max-width: 33.333333%;

}

}

</style>

<header class="container">

<!-- Шапка страницы -->

</header>

<main class="container">

<div class="row">

<article class="col col\_\_article">

<!-- Основная часть -->

</article>

<aside class="col col\_\_aside">

<!-- Боковая область -->

</aside>

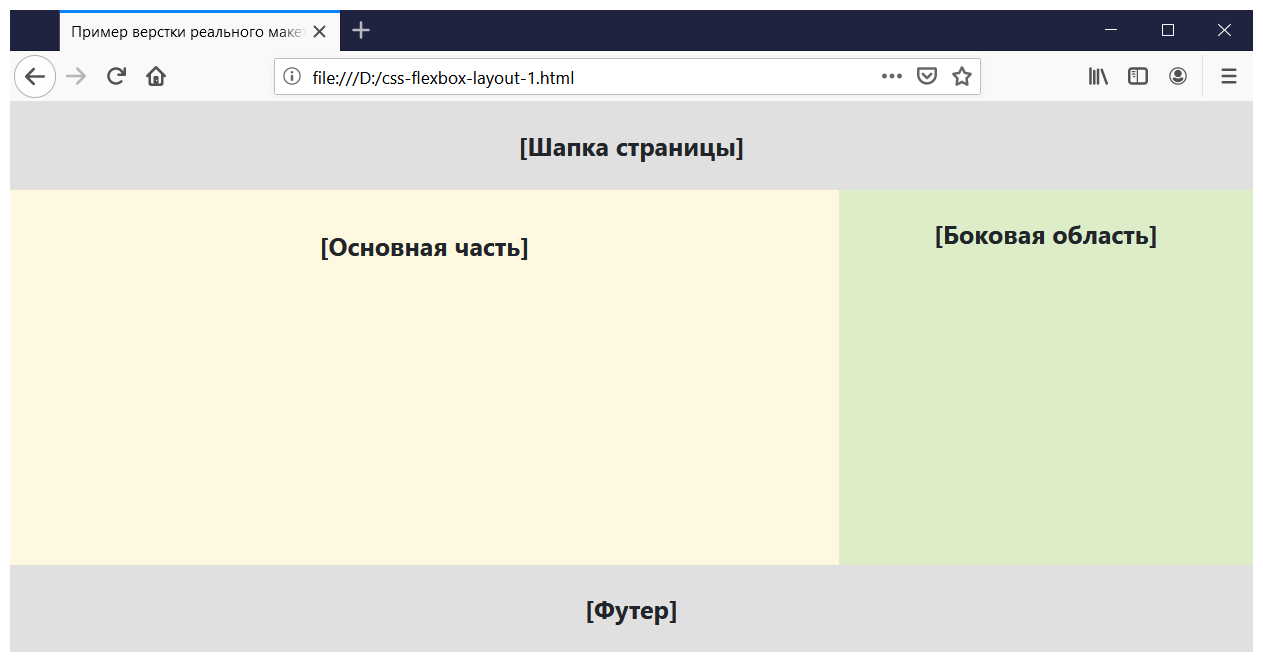
</div>

</main>

<footer class="container">

<!-- Футер -->

</footer>



В CSS для поддержки макета большинством браузеров добавлены свойства с [префиксами](https://itchief.ru/html-and-css/vendor-prefixes) и max-width.

Для «превращения» блока во flex-контейнер используется класс row. Установку ширины flex-элементам .col\_\_article и .col\_\_aside внутри flex-контейнера выполнено с использованием CSS-свойства flex.

В качестве примера разметим посредством flexbox ещё футер и создадим в элементе .col\_\_article блок состоящий из трёх элементов (минимальная ширина одного элемента - 300px). В футере разместим четыре блока (минимальная ширина одного блока - 200px).

<style>

/\* ... \*/

.col\_\_other-article {

-ms-flex: 1 0 0;

flex: 1 0 0;

min-width: 300px;

}

.col\_\_footer {

-ms-flex: 1 0 0;

flex: 1 0 0;

min-width: 200px;

}

</style>

<header class="container">

<!-- Шапка страницы -->

</header>

<main class="container">

<div class="row">

<article class="col col\_\_article">

<!-- Основная часть -->

<div class="row">

<div class="col col\_\_other-article">

<!-- Ещё 1 -->

</div>

<div class="col col\_\_other-article" style="padding-bottom: 15px;">

<!-- Ещё 2 -->

</div>

<div class="col col\_\_other-article" style="padding-bottom: 15px;">

<!-- Ещё 3 -->

</div>

</div>

</article>

<aside class="col col\_\_aside">

<!-- Боковая область -->

</aside>

</div>

</main>

<footer class="container">

<div class="row">

<div class="col col\_\_footer">

<!-- Секция футера 1 -->

</div>

</div>

<div class="row">

<div class="col col\_\_footer">

<!-- Секция футера 2 -->

</div>

</div>

<div class="row">

<div class="col col\_\_footer">

<!-- Секция футера 3 -->

</div>

</div>

<div class="row">

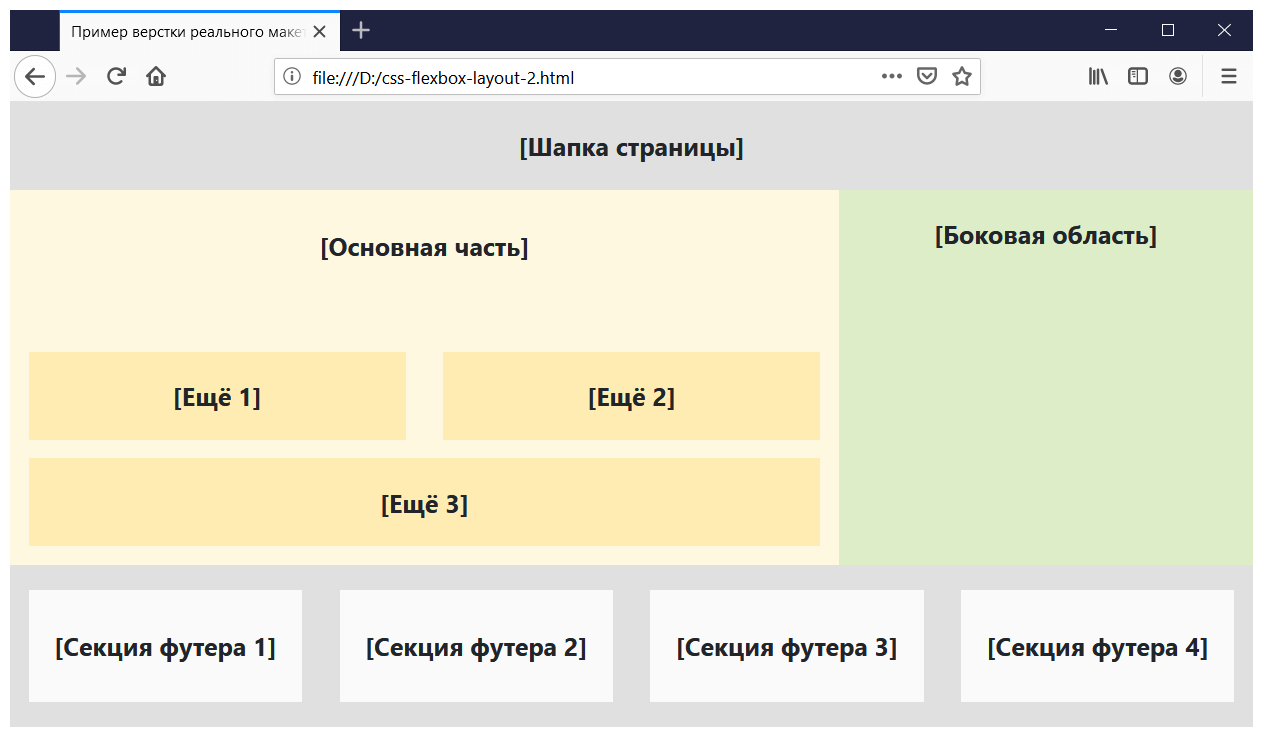
<div class="col col\_\_footer">

<!-- Секция футера 4 -->

</div>

</div>

</footer>



Если статья понравилась, то поделитесь ей в социальных сетях: