

CSS FLEXBOX

Flexbox разметка в CSS даёт один из наиболее эффективных способов расстановки, выравнивания и распределения места между элементами внутри контейнера, даже если их размер неизвестен или динамичен (собственно, по этому его и называют flex, от слова flexible, что по-английски имеет двойственное значение — гибкий и уступчивый, что очень сочетается с моделью поведения flexbox).

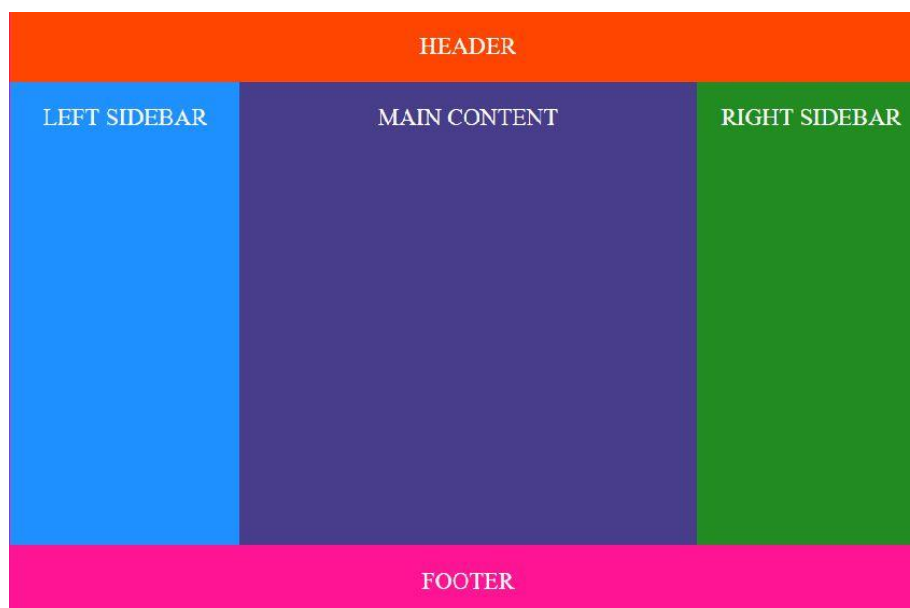
Основной целью flexbox является предоставление возможности изменения своих элементов по ширине и высоте, для того, чтобы они максимально эффективно умещались в доступном месте родительского контейнера, в частности — это удобно в тех случаях, когда нужно соответствовать всем типам дисплеев устройств и размерам экранов. Flex контейнер расширяет вложенные элементы для того, чтобы заполнить доступное пространство или же урезает их, чтобы избежать переполнения.

CSS flexbox (Flexible Box Layout Module) — модуль макета гибкого контейнера — представляет собой способ компоновки элементов, в основе лежит идея оси. Flexbox состоит из гибкого контейнера (flex container) и гибких элементов (flex items). Гибкие элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами.

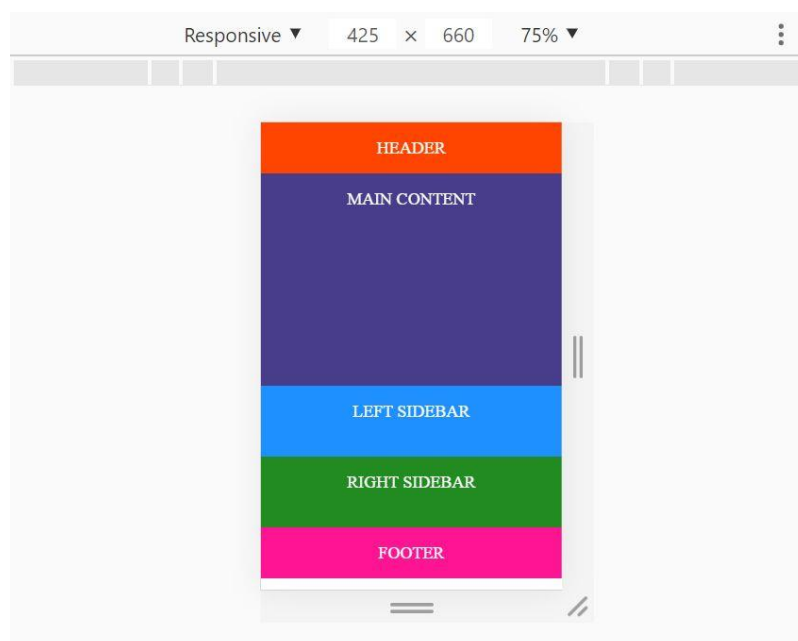
Модуль flexbox позволяет решать следующие задачи:

- Располагать элементы в одном из четырех направлений: слева направо, справа налево, сверху вниз или снизу вверх.
- Переопределять порядок отображения элементов.
- Автоматически определять размеры элементов таким образом, чтобы они вписывались в доступное пространство.
- Решать проблему с горизонтальным и вертикальным центрированием.
- Переносить элементы внутри контейнера, не допуская его переполнения.
- Создавать колонки одинаковой высоты.
- Создавать прижатый к низу страницы подвал (footer) сайта.

Рассмотрим, как создать макет с помощью flexbox. Во-первых, мы будем использовать один огромный flex-контейнер для всех элементов. Во-вторых, мы используем два меньших контейнера, которые будут разделять горизонтальные и вертикальные элементы.



И вот как это будет выглядеть на мобильных устройствах (помните, что это макет по умолчанию, который мы создадим первым):



У flexbox есть некоторые преимущества. Самое главное — он позволяет создавать контент-ориентированные боковые панели. Это связано с тем, что когда flexbox выделяет пространство для flex-элементов, он учитывает ширину их содержимого. В результате можно пропорционально распределить доступное пространство между основной областью и боковой панелью с помощью свойства flex.

Свойство `order` позволяет изменить порядок элементов в зависимости от размера области просмотра. Таким образом, на мобильных устройствах основная область будет находиться в верхней части экрана прямо под заголовком, а на больших экранах левая боковая панель будет находиться перед основной областью.

В файле HTML разместим область основного контента перед двумя боковыми панелями. Это связано прежде всего с тем, что согласно принципу «mobile-first» сначала будет выводиться основной контент для пользователей мобильных устройств. Однако этот порядок источников также улучшает доступность, так как программы чтения с экрана также будут читать основной контент перед боковыми панелями.

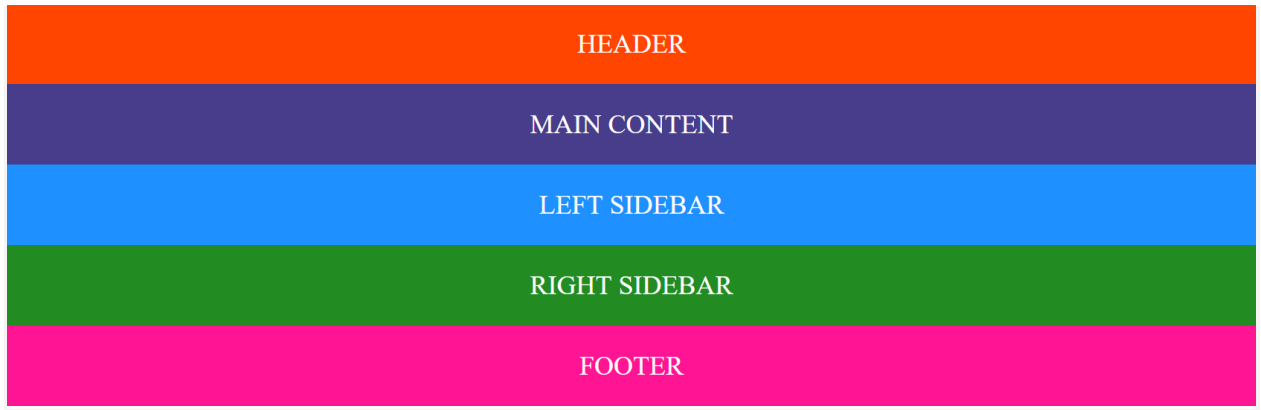
```
<div class="container">
  <header>HEADER</header>
  <main>MAIN CONTENT</main>
  <aside class="left-sidebar">LEFT SIDEBAR</aside>
  <aside class="right-sidebar">RIGHT SIDEBAR</aside>
  <footer>FOOTER</footer>
</div>
```

Основные стили

```
/* RESET */
* {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
}
/* COLORS and FONTS */
body {
  color: white;
  text-align: center;
  font-size: 24px;
  line-height: 3;
}
header {
  background: orangered;
}
main {
  background: darkslateblue;
}
footer {
  background: deeppink;
}
.left-sidebar {
  background: dodgerblue;
}
.right-sidebar {
```

```
background: forestgreen;
}
```

Получим результат



Добавим правила FLEXBOX для мобильного представления. Сначала мы устанавливаем для свойства `display` — `flex` и разместим flex-элементы вертикально друг над другом, используя `flex-direction: column;`.

```
.container {

    display: flex;

    flex-direction: column;

}
```

Добавим правила FLEXBOX для настольного представления, а именно, **медиа-запрос `min-width`**. Прежде всего заменим значение `flex-direction` на `row`, так как теперь основная область и две боковые панели будут отображаться рядом друг с другом. Установим для `flex-wrap` значение `wrap` так, чтобы flex-элементы могли красиво переноситься вдоль главной оси flex-контейнера. Установим для `width` элементов `header` и `footer` `100%`, так как они должны растягиваться на весь экран. Кроме того, используем правило `min-height: 80vh;` для раздела `main`, так что демо будет иметь высоту по умолчанию даже без содержимого.

```
@media all and (min-width: 768px) {
    .container {
        flex-direction: row;
```

```

        flex-wrap: wrap;
    }
    header,
    footer {
        width: 100%;
    }
    main {
        flex: 2;
        order: 2;
        min-height: 80vh;
    }

    .left-sidebar {
        order: 1;
        flex: 1;
    }
    .right-sidebar {
        flex: 1;
        order: 3;
    }
    footer {
        order: 4;
    }
}

```

Переупорядочим элементы, используя свойство `order`. Таким образом, левая боковая панель для пользователей настольных компьютеров будет отображаться перед основной областью. Значение по умолчанию для `order` равно 0 для каждого элемента. Таким образом, не нужно менять его для первого элемента (`header`), так как оно должно быть равным 0. Тем не менее, мы присваиваем индивидуальное значение `order` всем остальным элементам, от 1 до 4.

Свойство `flex` определяет, как доступное пространство распределяется между flex-элементами. Поскольку элементы `header` и `footer` имеют фиксированную ширину (100% экрана), оставшееся пространство распределяется только между основной областью и двумя боковыми панелями в пропорции 2: 1: 1.

```

<html>
<head>
<style>
/* RESET */
* {
    box-sizing: border-box;
    padding: 0;
    margin: 0;
}
/* COLORS and FONTS */
body {
    color: white;

```

```
        text-align: center;
        font-size: 24px;
        line-height: 3;
    }
    header {
        background: orangered;
    }
    main {
        background: darkslateblue;
    }
    footer {
        background: deeppink;
    }
    .left-sidebar {
        background: dodgerblue;
    }
    .right-sidebar {
        background: forestgreen;
    }

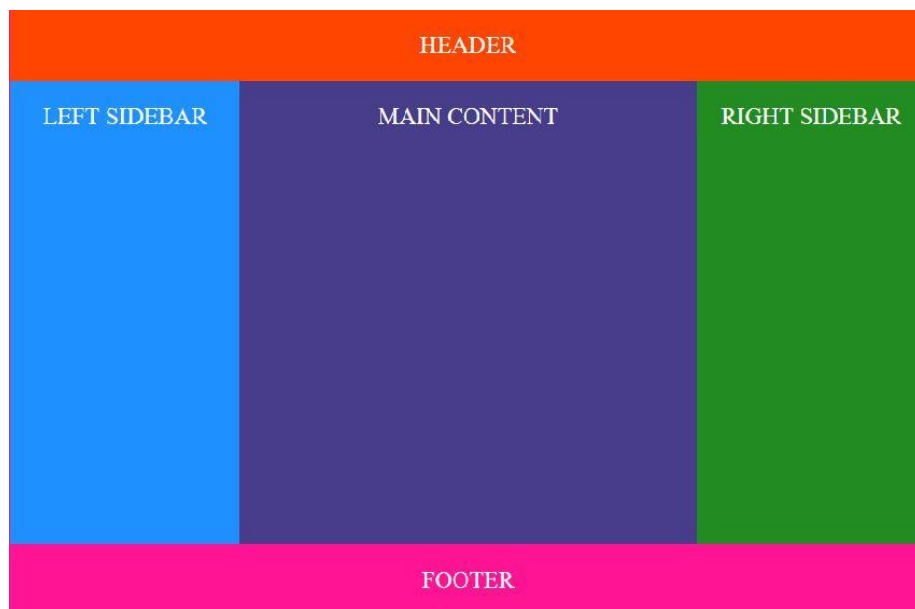
    .container {
        display: flex;
        flex-direction: column;
    }
    @media all and (min-width: 768px) {
        .container {
            flex-direction: row;
            flex-wrap: wrap;
        }
        header,
        footer {
            width: 100%;
        }
        main {
            flex: 2;
            order: 2;
            min-height: 80vh;
        }

        .left-sidebar {
            order: 1;
            flex: 1;
        }
        .right-sidebar {
            flex: 1;
            order: 3;
        }
        footer {
            order: 4;
        }
    }
</style>
```

```
</head>

<body>
<div class="container">
  <header>HEADER</header>
  <main>MAIN CONTENT</main>
  <aside class="left-sidebar">LEFT SIDEBAR</aside>
  <aside class="right-sidebar">RIGHT SIDEBAR</aside>
  <footer>FOOTER</footer>
</div>
</body>
</html>
```

Результат



Более подробно про flex-разметку можно прочитать, например, [Как работает CSS Flexbox: наглядное введение в систему компоновки элементов на веб-странице \(tproger.ru\)](http://tproger.ru)

CSS GRID

CSS Grid это новая модель для создания шаблонов, оптимизированная для создания двумерных макетов. Она идеально подходит для: *шаблонов сайтов, форм, галерей и всего, что требует точного и отзывчивого позиционирования.*

Grid шаблон работает по системе сеток. Grid это набор пересекающихся горизонтальных и вертикальных линий, которые создают размерность и позиционируют систему координат для контента в самом grid-контейнере.

Чтобы создать Grid разметку, вам просто нужно выставить элементу `display: grid`. Этот шаг автоматически сделает всех прямых потомков этого элемента — grid элементами. После этого можно использовать разнообразные grid свойства для выравнивания размеров и позиционирования элементов должным образом.

Рассмотрим предыдущий пример с применением CSS Grid.

Определим области сетки, к которым мы можем обратиться через псевдоним. Делается это при помощи свойства `grid-area`.

```
.gr_header { grid-area: header; background: orangered; }
.gr_footer { grid-area: footer; background: deeppink; }
.gr_main { grid-area: main; background: darkslateblue; }
.gr_left { grid-area: left; background: dodgerblue }
.gr_right { grid-area: right; background: forestgreen; }
```

Затем, используя свойство `grid-template-areas` мы можем расположить элементы на сетке интуитивным и визуальным способом. Свойство `grid-template-areas` принимает список из строк разделенных пробелами. Каждая строчка представляет собой ряд. В каждой строке, у нас есть список областей сетки разделенных пробелами. Каждая область сетки занимает один столбец. Так что, если мы хотим, чтобы область охватила два столбца мы определяем ее дважды.

В макете у нас есть 3 столбца и 3 ряда. Header и footer занимают 3 колонки, в то время как другие области охватывают по 1 колонке каждый.

```
.gr {
  display: grid;
  grid-template-areas: "header header header"
                      "left main right"
                      "footer footer footer";
}
```


Далее, мы хотим определить ширину столбцов. Она определяется при помощи свойства `grid-template-columns`. Это свойство принимает разделенный пробелами список. Поскольку у нас 3 колонки, то и ширину мы определяем 3 раза.

```
grid-template-columns: [column 1 width] [column 2 width] [column 3 width];
```

В макете мы хотим видеть 2 сайдбара по 150 пикселей каждый.

```
.gr { grid-template-columns: 150px [column 2 width] 150px;}
```

Также мы хотим, чтобы средний столбец занимал оставшуюся часть пространства. Мы можем сделать это при помощи новой единицы измерения `fr`. Она обозначает долю свободного пространства в сетке. В нашем случае добавляется еще и ширина сайдбаров, в сумме 300px.

```
.gr {  
  grid-template-columns: 150px 1fr 150px;}
```

Теперь нам нужно определить высоту рядов. Подобно тому, как мы определяем ширину столбцов при помощи `grid-template-columns`, мы определяем высоту при помощи `grid-template-rows`. Это свойство принимает разделенный пробелами список содержащий высоту для каждого ряда в нашей сетке. Хотя мы можем записать его на одной строке или написать по одному ряду в строку.

```
.hg {  
  grid-template-rows: 100px  
                     1fr  
                     30px;}
```

Таким образом, высота header будет равняться 100px, высота footer 30, а средний ряд (основное содержимое и две боковые панели) займет оставшуюся свободную часть.

В данном макете мы хотим, чтобы футер всегда находился в нижней части экрана, даже если содержимого на странице мало. Для этого установим минимальную высоту элемента `.gr`. Поскольку мы указали, что средний ряд должен занимать оставшуюся часть свободного пространства он растягивается и заполняет экран.

```
.hg {  
  min-height: 100vh;}
```

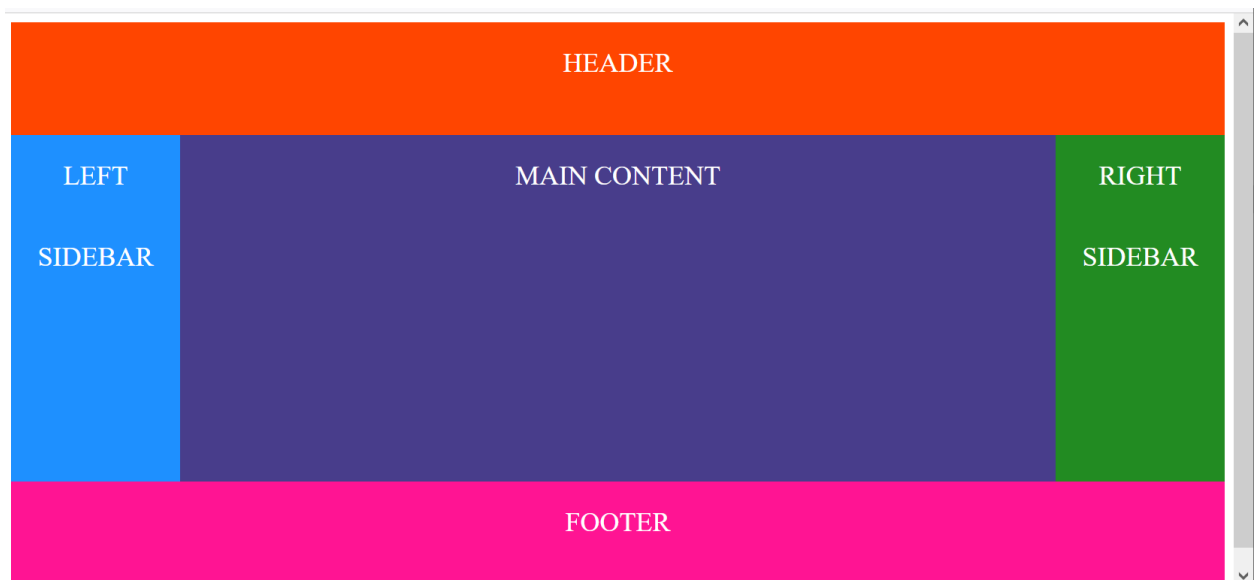
Приёмы отзывчивого (responsive) веб-дизайна базируются на использовании процентных значений. Но проценты далеко не лучшее решение для каждого случая, так как они вычисляются относительно размеров ближайшего родительского элемента. Поэтому, если вы хотите использовать высоту и ширину окна браузера, лучше воспользоваться единицами `vh` и `vw`. Например, если высота окна браузера равна 900px, то `1vh` будет равен 9px. Аналогично, если ширина окна браузера равна 1600px, то `1vw` будет равен 16px. Так как ширина элемента, указанная с помощью `100vw` больше ширины области просмотра, то для создания полноэкранных фоновых изображений лучше использовать ширину `100%`, которая будет равна ширине корневого элемента `html`.

И, наконец, мы хотим сделать макет отзывчивым. На небольших устройствах все элементы сетки должны отображаться в одном столбце, один за другим. Для этого нам необходимо переопределить 3 свойства, которые мы определили ранее `grid-template-areas`, `grid-template-columns` и `grid-template-rows`.

Во-первых, мы хотим чтобы все элементы в сетке были в одном столбце в определенном порядке. Далее, мы хотим чтобы все элементы растянулись на всю ширину сетки. И наконец, нам нужно сбросить высоту для каждой из строк. Все строки, кроме основного ряда, имеют определенную высоту.

```
@media screen and (max-width: 600px) {  
  .gr {  
    grid-template-areas: "header"  
                        "left"  
                        "main"  
                        "right"  
                        "footer";  
    grid-template-columns: 100%;  
    grid-template-rows: 100px  
                      50px  
                      1fr  
                      50px  
                      30px;  
  }
```

Результат



Код целиком

```
<html>
<head>
<style>
body {
    color: white;
    text-align: center;
    font-size: 24px;
    line-height: 3;
}

.gr_header { grid-area: header; background: orangered;}
.gr_footer { grid-area: footer; background: deeppink;}
.gr_main { grid-area: main; background: darkslateblue;}
.gr_left { grid-area: left; background: dodgerblue}
.gr_right { grid-area: right; background: forestgreen;}

.gr {
    display: grid;
    grid-template-areas: "header header header"
                        "left main right"
                        "footer footer footer";
    grid-template-columns: 150px 1fr 150px;
    grid-template-rows: 100px
                      1fr

```

```

        100px;
    min-height: 100vh;
}

@media screen and (max-width: 600px) {
    .gr {
        grid-template-areas: "header"
                             "left"
                             "main"
                             "right"
                             "footer";

        grid-template-columns: 100%;
        grid-template-rows: 100px
                             50px
                             1fr
                             50px
                             30px;
    }
}
</style>

</head>

<body class="gr">
    <header class="gr_header">HEADER</header>
    <main class="gr_main">MAIN CONTENT</main>
    <aside class="gr_left">LEFT SIDEBAR</aside>
    <aside class="gr_right">RIGHT SIDEBAR</aside>
    <footer class="gr_footer">FOOTER</footer>
</body>
</html>

```

Более подробно можно почитать по Grid, например, [7 основных понятий CSS Grid Layout с примерами, которые помогут начать работу с гридами \(tproger.ru\)](#)

ЗАДАНИЕ


Используя *CSS Grid Layout* и *CSS Flexbox Layout* реализовать разметку страницы (тема любая). Навигация по странице осуществляется с помощью ссылок-якорей. Пример фиксированного меню приведен ниже.

COFFEE HOUSE

HOMESTORYBLOGSUBSCRIBE

TAKE YOUR TIME


STORY



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean varius eget arcu quis cursus. Nunc id tortor imperdiet, vestibulum orci sed, gravida orci. Cras a enim ac elit eleifend euismod. Class aptent taciti sociosq ad litora torquent per conubia nostra, per inceptos himenaeos. Praesent mattis ultrices est, congue ullamcorper erat bibendum pulvinar. In accumsan dapibus accumsan. Maecenas semper est ut nunc feugiat, vel ultrices ligula commodo. Donec eget pulvinar nunc, eget pharetra risus.


BLOG

Daily Coffee News




Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean varius eget arcu quis cursus. Nunc id tortor imperdiet, vestibulum orci sed, gravida orci. Cras a enim ac elit eleifend euismod.

Dutch Coffee Chains



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean varius eget arcu quis cursus. Nunc id tortor imperdiet, vestibulum orci sed, gravida orci. Cras a enim ac elit eleifend euismod.

Start Up In Minneapolis



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean varius eget arcu quis cursus. Nunc id tortor imperdiet, vestibulum orci sed, gravida orci. Cras a enim ac elit eleifend euismod.

SUBSCRIBE

enter your email

SUBMIT

Заголовок 1 уровня

Заголовок 2 уровня

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

текст текст текст текст текст текст текст текст ...

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {margin:0;}

.navbar {
  overflow: hidden;
  background-color: #333;
  position: fixed;
  top: 0;
  width: 100%;
}

.navbar a {
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

.navbar a:hover {
  background: #ddd;
  color: black;
}

.topnav-right {
  float: right;
}

.main {
  padding: 16px;
  margin-top: 30px;
  height: 1500px;
}
</style>
```

```
</head>
<body>

<div class="navbar">
  <a href="#home">Home</a>
  <a href="#news">News</a>
  <a href="#contact">Contact</a>

<div class="topnav-right">
  <a href="#search">Search</a>
  <a href="#about">About</a>
</div>
</div>

<div class="main">

  <h1>Заголовок 1 уровня</h1>
  <h2>Заголовок 2 уровня</h2>

  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>
  <p>текст текст текст текст текст текст текст текст ...</p>

</div>

</body>
</html>
```