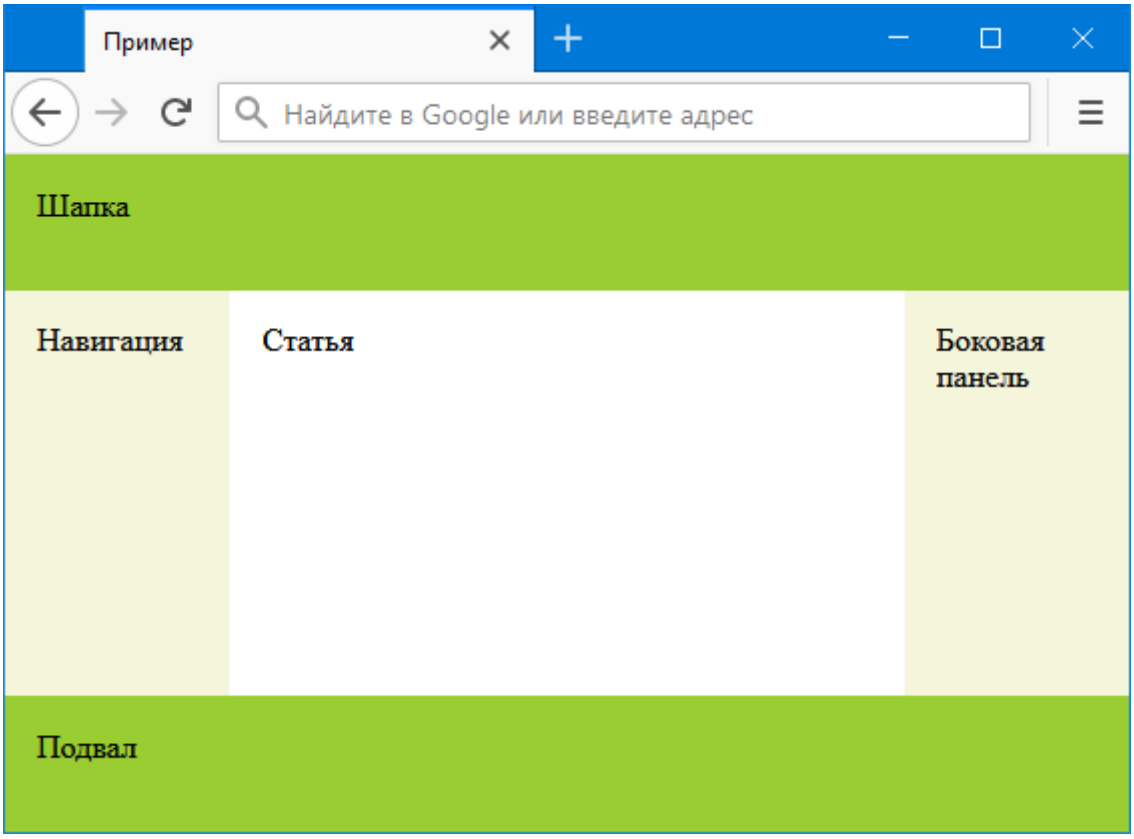
Макет сайта на флексбоксах

Флексбоксы идеально подходят для создания широко используемых макетов веб-сайтов, вроде трёхколоночного, так называемый макет «Святой Грааль», где все колонки должны занимать полную высоту, независимо от их содержимого. При этом в исходном коде основное содержимое идёт до навигации, а на самой странице основное содержимое идёт после навигации.

Вроде этого.

☐ Тёмная тема

```
<!doctype html>
<title>Пример</title>
<style>
* {
  box-sizing: border-box;
}
body {
  margin: 0;
}
#main {
  display: flex;
  min-height: calc(100vh - 40vh);
}
#main > article {
  flex: 1;
}
#main > nav,
#main > aside {
  flex: 0 0 20vw;
  background: beige;
}
#main > nav {
  order: -1;
}
header, footer, article, nav, aside {
  padding: 1em;
}
header, footer {
  background: yellowgreen;
  height: 20vh;
}
</style>
<body>
  <header>Шапка</header>
  <div id="main">
    <article>Статья</article>
    <nav>Навигация</nav>
    <aside>Боковая панель</aside>
  </div>
  <footer>Подвал</footer>
</body>
```



До появления флексбоксов такой макет было довольно трудно получить без использования каких-либо хаков. Разработчикам часто приходилось делать такие вещи, как добавление дополнительной разметки, применение отрицательных margin и другие трюки, чтобы всё правильно выстраивалось независимо от объёма содержимого или размера экрана. Но, как показывает приведённый выше пример, на флексбоксах всё получается гораздо проще.

Вот краткое изложение кода. В данном примере мы делаем элемент #main флекс-контейнером, а шапку и подвал оставляем блочными элементами. Другими словами, флексбоксом становится только средняя часть. Вот фрагмент, который делает её флекс-контейнером.

```
#main {
  display: flex;
  min-height: calc(100vh - 40vh);
}
```

Просто используем display: flex, чтобы сделать флекс-контейнер. Обратите внимание, мы также задаём значение min-height с помощью функции calc() и устанавливаем #main как 100% от высоты области просмотра *минус* высота шапки и подвала (по 20vh каждый). Это гарантирует, что макет будет занимать всю высоту экрана, даже если в нём мало содержимого. В итоге подвал никогда не поднимется и не оставит пустого пространства под ним, если содержимое занимает меньше высоты экрана.

Вычисление min-height было достаточно простым, учитывая, что мы применили box-sizing: border-box ко всем элементам. Если бы мы этого не сделали, тогда нужно было бы добавить значение padding к сумме для вычитания.

После установки флекс-контейнера мы имеем дело с флекс-элементами.

```
#main > article {
  flex: 1;
}
#main > nav,
#main > aside {
  flex: 0 0 20vw;
  background: beige;
}
#main > nav {
```

```
  order: -1;
}
```

Свойство flex является сокращением для свойств flex-grow, flex-shrink и flex-basis. В первом случае мы написали только одно значение, поэтому flex устанавливает свойство flex-grow. Во втором случае мы написали все три значения для <nav> и <aside>, чтобы установить нулевое значение flex-grow, нулевое значение flex-shrink и значение 20vw для flex-basis. Свойство flex-basisустанавливает исходный основной размер флекс-элемента до распределения свободного пространства. Таким образом, мы просто устанавливаем ширину <nav> и <aside>, а также их цвет фона.

Вот и весь код, благодаря которому макет работает.

В этом примере флексбоксы применяются только для центрального раздела, где располагается основное содержимое. Если вы желаете использовать флексбоксы для всей страницы, то можете добавить вложенные флекс-контейнеры.