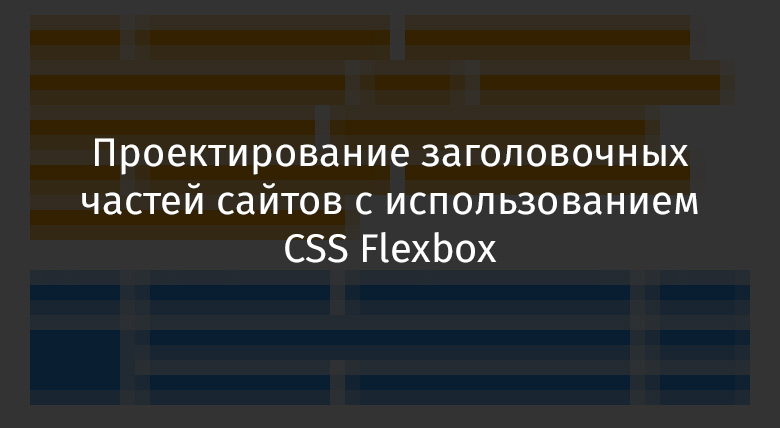
Проектирование заголовочных частей сайтов с использованием CSS Flexbox

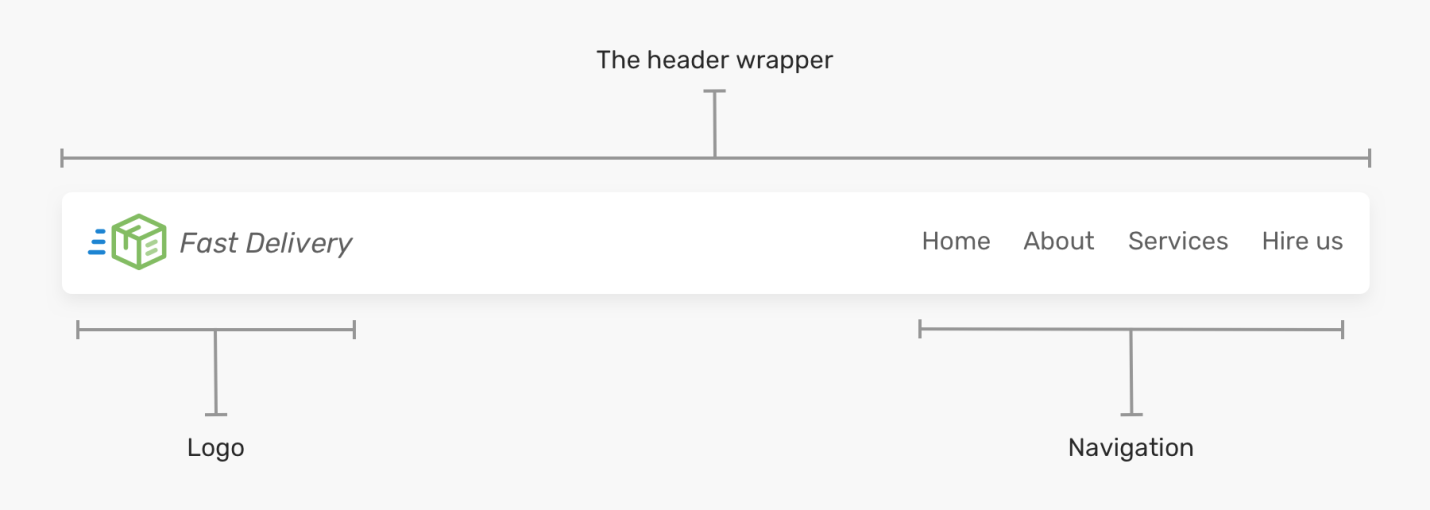
[Автор оригинала: Ahmad Shadeed](https://ishadeed.com/article/website-headers-flexbox/)

* [Блог компании RUVDS.com](https://habr.com/ru/company/ruvds/),
* [Разработка веб-сайтов](https://habr.com/ru/hub/webdev/),
* [CSS](https://habr.com/ru/hub/css/),
* [HTML](https://habr.com/ru/hub/html5/)
* Перевод

Когда я, изучив основы HTML и CSS в 2014 году, занялся разработкой сайтов, сложнее и страшнее всего для меня было создание их заголовочных частей. Тогда технология Flexbox была ещё достаточно новой, поэтому мы вынуждены были использовать старые методы — вроде float-позиционирования и техники clearfix. Сегодня мир фронтенд разработки стал совсем другим. А именно, технология Flexbox получила широкую браузерную поддержку, что открывает нам множество новых возможностей.  
  
[](https://habr.com/ru/company/ruvds/blog/518514/)  
  
Кто-то может сказать, что сегодня, из-за того, что у нас есть современные CSS-технологии разработки макетов страниц, создать заголовочную часть сайта проще, чем в былые времена. Но, на самом деле, это не так. И в наши дни при проектировании макетов страниц приходится решать непростые и интересные задачи. О некоторых из них я собираюсь рассказать в этой статье.  
  
А именно, здесь я расскажу о том, как, используя Flexbox-макеты, создавать качественные заголовочные части сайтов. Я поделюсь некоторыми советами, а в итоге продемонстрирую мой проект, созданный специально для этого материала. Поэтому, если вам и правда интересна тема, которую я тут решил поднять, вам, определённо, стоит дочитать эту статью до конца.  
  
Здесь я исхожу из предположения о том, что вы владеете основами Flexbox-вёрстки. [Вот](https://habr.com/ru/company/ruvds/blog/515298/), на всякий случая, моя статья, написанная для тех, кто хочет узнать больше о CSS-свойстве flex.

Введение

Для начала давайте удостоверимся в том, что мы с вами называем «заголовочной частью сайта» или «заголовком сайта» одно и то же. Заголовочная часть сайта — это один из первых элементов страницы, которые пользователь видит, посещая сайт. Обычно она содержит логотип или название сайта, а также — навигационные ссылки. Взгляните на следующий рисунок.



*Логотип сайта, навигационные ссылки и контейнер заголовочной части сайта*  
  
Вне зависимости от дизайна заголовочной части сайта её ключевыми элементами является логотип и список ссылок.

Технология Flexbox в действии

Когда макет для заголовочной части сайта создают с использованием технологии Flexbox, все элементы одного уровня выстраиваются в одну строку. Затем остаётся лишь воспользоваться свойством justify-content для выравнивания элементов путём распределения свободного пространства, имеющегося между ними.  
  
Вот код разметки:

<header class="site-header">

  <a href="#" class="brand">Brand</a>

  <nav class="nav"></nav>

</header>

Вот стили:

.site-header {

  display: flex;

  justify-content: space-between;

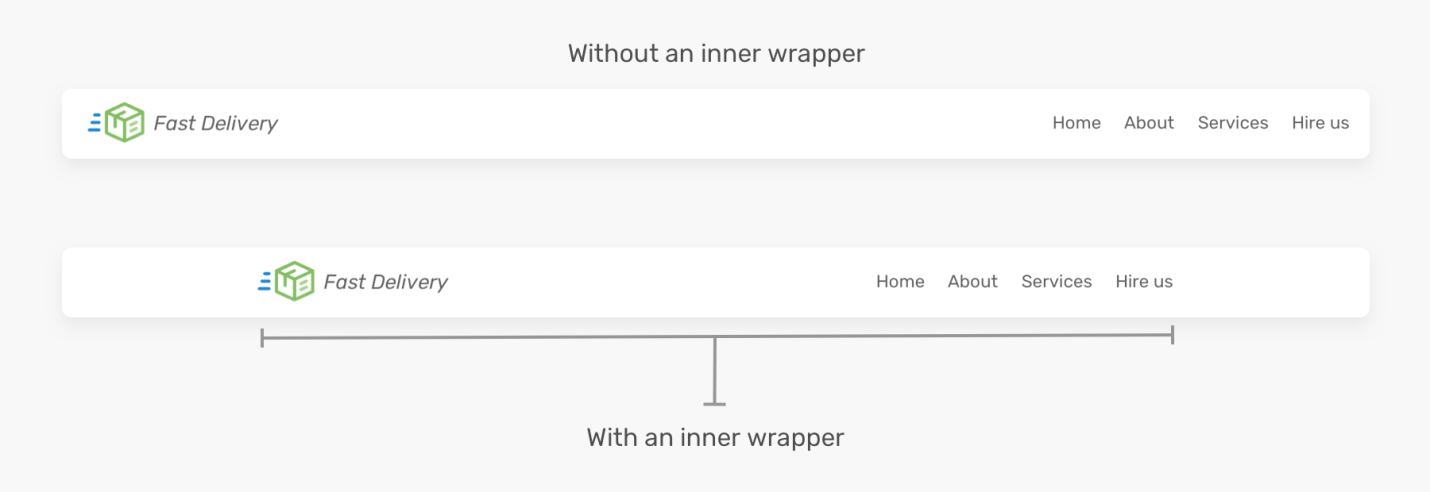
  align-items: center;

}

Всё совсем несложно, правда? Ну, для столь простого заголовка — так оно и есть. Но на практике всё может оказаться куда сложнее.

Контейнер заголовочной части страницы

В коде разметки заголовочной части страницы, описанной в предыдущем разделе, нет внутреннего элемента-контейнера, в который помещены элементы, представляющие логотип и навигационные ссылки. На больших экранах применение подобного макета может привести к возникновению проблем.



*Макет заголовочной части страницы, созданный без использования внутреннего контейнера (выше) и с использованием контейнера (ниже)*  
  
Обратите внимание на то, что первый заголовок страницы оказывается слишком широким из-за того, что при его создании не применялся [внутренний контейнер](https://ishadeed.com/article/styling-wrappers-css/). А вот второй макет выглядит уже гораздо лучше. Поэтому HTML-код предыдущего примера стоит переписать так:

<header class="site-header">

  <div class="wrapper site-header\_\_wrapper">

    <a href="#" class="brand"><img src="logo.svg" alt="brand" /></a>

    <nav class="nav"></nav>

  </div>

</header>

А flex-стили нужно применить к элементу .site-header\_\_wrapper:

.site-header\_\_wrapper {

  display: flex;

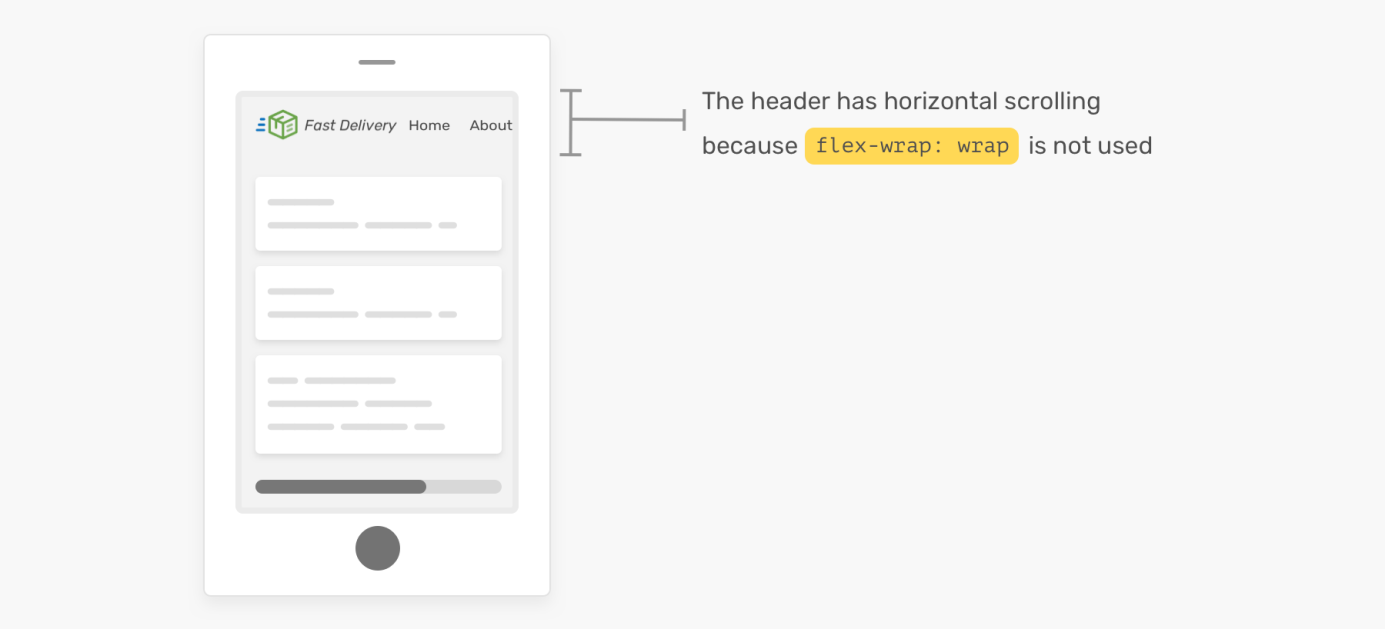
  justify-content: space-between;

  align-items: center;

}

Использование свойства flex-wrap

Свойство flex-wrap — это нечто вроде защитного CSS-механизма. Если размеры экрана невелики, есть вероятность того, что для работы с заголовочной частью страницы придётся пользоваться горизонтальной прокруткой. Вот как это выглядит.

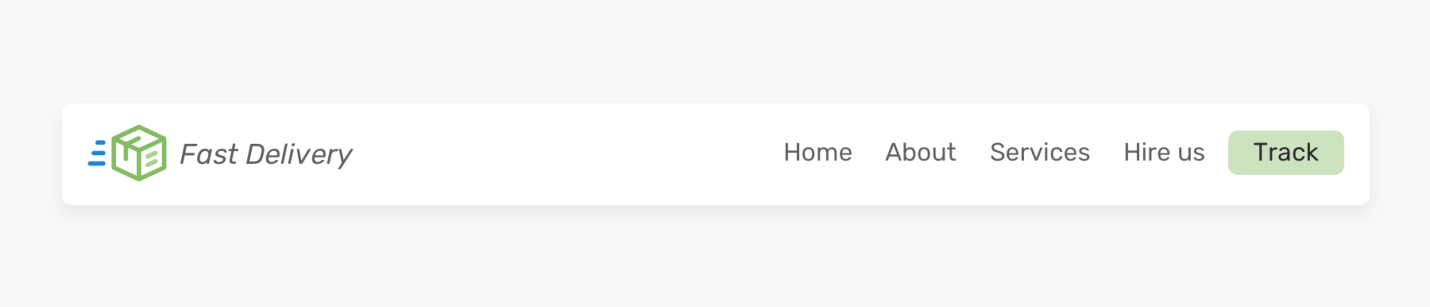


*Для работы с заголовком страницы необходимо применять горизонтальную прокрутку. А всё дело в том, что тут не используется свойство flex-wrap: wrap*  
  
Как видите, работать с такой страницей будет неудобно. Поэтому не забывайте о свойстве flex-wrap!

Исследование различных вариантов макетов заголовочных частей страниц

Во Flexbox-макетах мне нравится то, что они упрощают поддержку различных вариантов дизайна заголовочных частей страниц. В этом разделе я, основываясь на вышеприведённом примере, собираюсь исследовать возможности по расширению макета путём добавления в него новых элементов, вроде кнопок и полей поиска. Тут же я хочу изучить способы изменения порядка вывода дочерних элементов заголовочной части страницы.

▍Вариант №1



*Первый вариант заголовочной части страницы*  
  
В этот макет была добавлена кнопка, размещённая после блока навигационных ссылок. Как это сделать? Нужно ли оформить её в виде ссылки, размещённой внутри элемента <nav>? А, может, её следует сделать отдельным элементом? Я поступлю именно так.  
  
Вот HTML-код, выражающий мою идею:

<header class="site-header">

  <div class="wrapper site-header\_\_wrapper">

    <a href="#" class="brand"><img src="logo.svg" alt="brand" /></a>

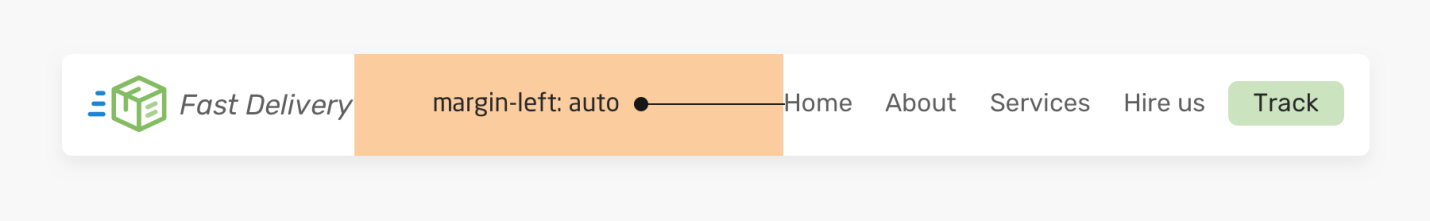
    <nav class="nav"></nav>

    <a href="/track-shipment" class="button">Track</a>

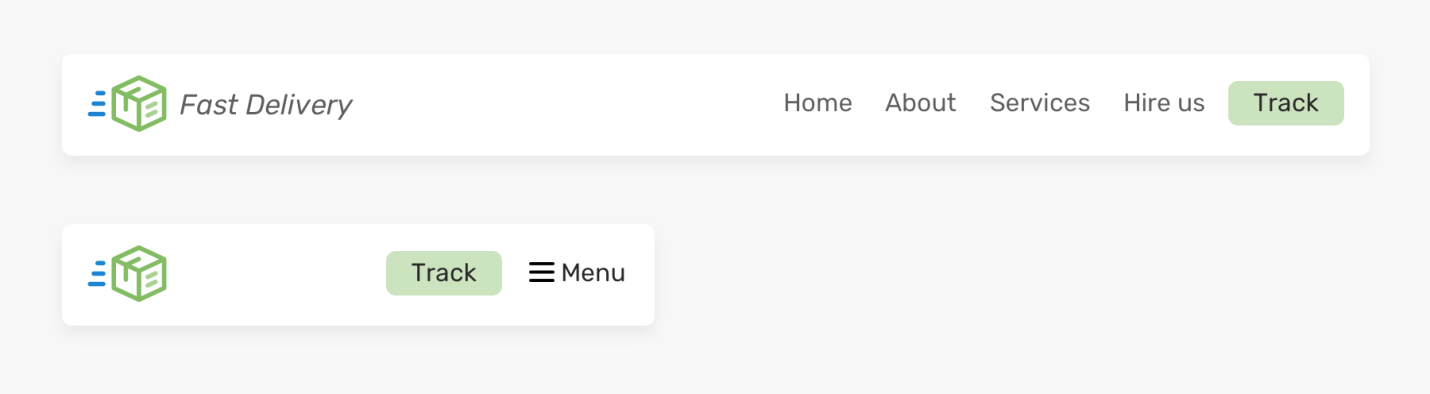
  </div>

</header>

В данном случае расстояние между элементами не получится настроить с помощью свойства justify-content: space-between. Вместо этого я, для выравнивания блока ссылок, воспользуюсь свойством margin-left: auto. Благодаря этому ссылки и кнопки окажутся прижатыми к правой части контейнера.

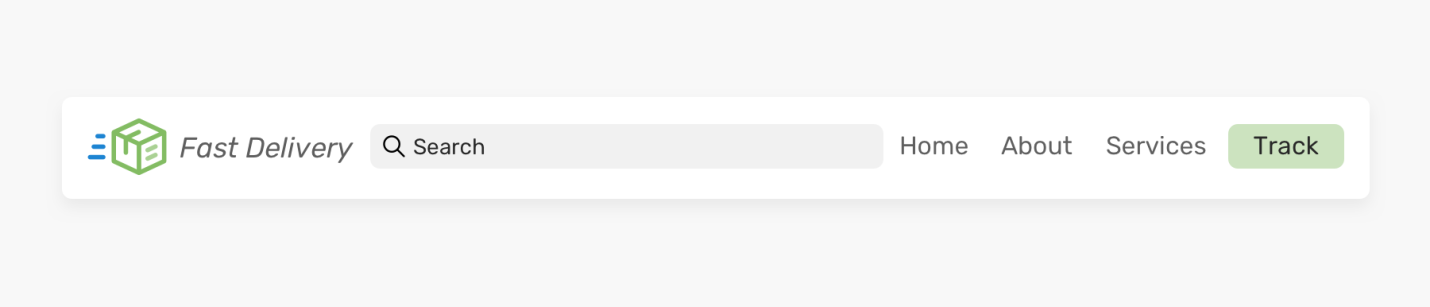


*Элементы, благодаря margin-left: auto, прижаты к правой части контейнера*  
  
Отделение кнопки от блока ссылок ценно для мобильных макетов, где, например, может понадобиться, оставив кнопку, представить блок ссылок в виде выпадающего меню.



*Заголовочная часть страницы на настольном и на мобильном устройстве*

▍Вариант №2



*Второй вариант заголовочной части страницы*  
  
Тут мы расширили возможности заголовочной части страницы, добавив на неё поле поиска. Оно занимает свободное место, которое осталось после размещения на странице других элементов. С использованием Flexbox-макетов этого можно достичь, применив свойство flex.  
  
Вот разметка:

<header class="site-header">

  <div class="wrapper site-header\_\_wrapper">

    <a href="#" class="brand"><img src="logo.svg" alt="brand" /></a>

    <div class="search"></div>

    <nav class="nav"></nav>

    <a href="/track-shipment" class="button">Track</a>

  </div>

</header>

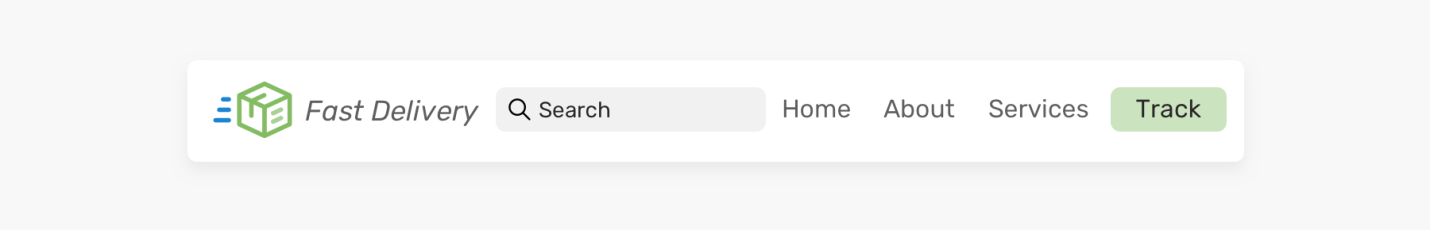
Вот — стиль, применяемый к полю поиска:

.search {

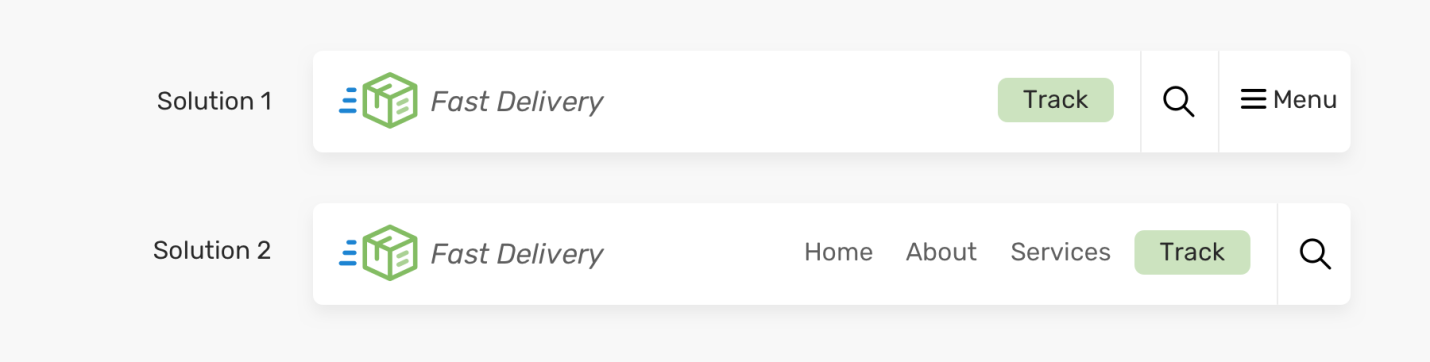
  flex: 1;

}

Вот и всё! Теперь поле заполнит пустое место, которое имеется между логотипом и блоком ссылок. Правда, у такого подхода есть некоторые ограничения. На маленьких экранах заголовочная часть страницы будет выглядеть так, как показано ниже.



*Заголовочная часть страницы на маленьком экране*  
  
Ширина поля поиска не должна быть меньше определённого размера. Иначе в него будет сложно вводить текст. Ниже приведено несколько вариантов решения этой проблемы.



*Решение проблемы, вызванной уменьшением ширины поля поиска на небольших экранах*  
  
Мне больше нравится второй вариант, так как при его использовании блок ссылок не подвергается слишком раннему скрытию. В целом, могу отметить, что я стремлюсь к тому, чтобы не скрывать элемент в том случае, если он не мешает правильному выводу макета.

▍Вариант №3



*Третий вариант заголовочной части страницыы*  
  
Здесь используется та же разметка, что и в первом варианте заголовка. Но тут изменён порядок вывода элементов на экран. Как это сделать? Возможно, вы решите, что тут стоит воспользоваться свойством order. Изучим эту идею.  
  
Вот HTML-код:

<header class="site-header">

  <div class="wrapper site-header\_\_wrapper">

    <a href="#" class="brand"><img src="logo.svg" alt="brand" /></a>

    <nav class="nav"></nav>

    <a href="/track-shipment" class="button">Track</a>

  </div>

</header>

Вот стили:

.site-header {

  display: flex;

  justify-content: space-between;

}

.nav {

  order: -1;

}

А вот как выглядит результат визуализации всего этого.



*Использование свойства justify-content: space-between не приводит к выравниванию логотипа по центру. Благодаря этому свойству лишь распределяется свободное пространство между элементами*  
  
Решение этой проблемы заключается в назначении каждому дочернему элементу заголовочной части страницы свойства flex: 1. Благодаря этому пространство будет распределено между элементами.  
  
Вот стили:

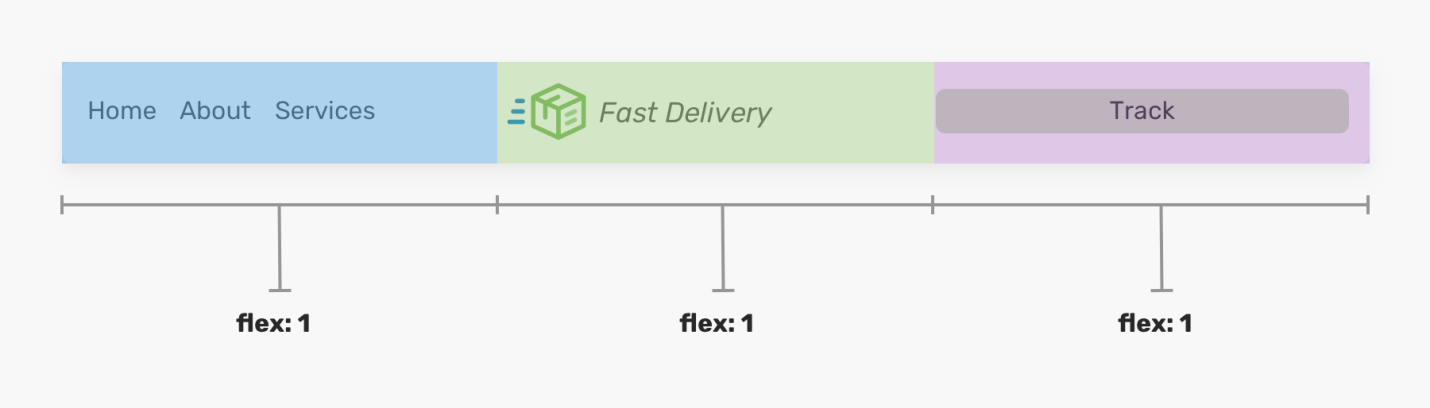
.brand,

.nav,

.button {

  flex: 1;

}



*Равномерное распределение свободного пространства между элементами заголовочной части страницы*  
  
Правда, тут с кнопкой произошла какая-то странность. Она, из-за применения свойства flex: 1, стала слишком длинной. Единственный способ это исправить заключается в том, чтобы поместить кнопку в элемент-контейнер:

<header class="site-header">

  <div class="wrapper site-header\_\_wrapper">

    <a href="#" class="brand"><img src="logo.svg" alt="brand" /></a>

    <nav class="nav"></nav>

    <div class="button-wrapper">

      <a href="/track-shipment" class="button">Track</a>

    </div>

  </div>

</header>

При таком подходе можно выровнять по центру и логотип, и кнопку:

.logo {

  text-align: center;

}

*/\* Не обращайте внимания на именование сущностей. Это - лишь демонстрация, хотя я понимаю то, что и в подобных примерах стоило бы лучше следить за именами. \*/*

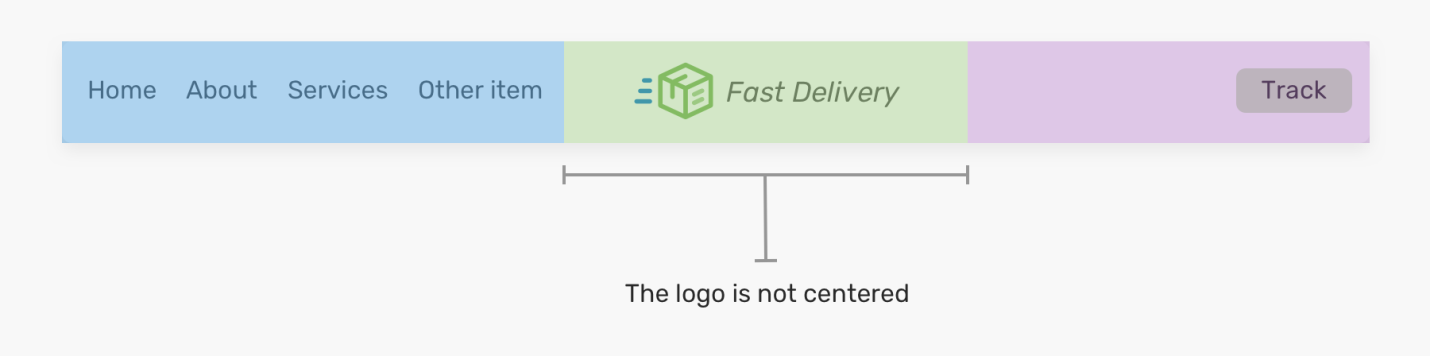
.button-wrapper {

  text-align: end; */\* end - это, в LTR-языках, то же самое, что и right \*/*

}



*Выравнивание элементов*  
  
Этот подход, правда, тоже не идеален. Помните о том, что он, если в навигационном блоке будет больше ссылок, приведёт к появлению проблем. А именно, тут нужно, чтобы ширина навигационного блока не превышала бы определённого размера. Вот пример, в котором логотип теряет выравнивание по центру.

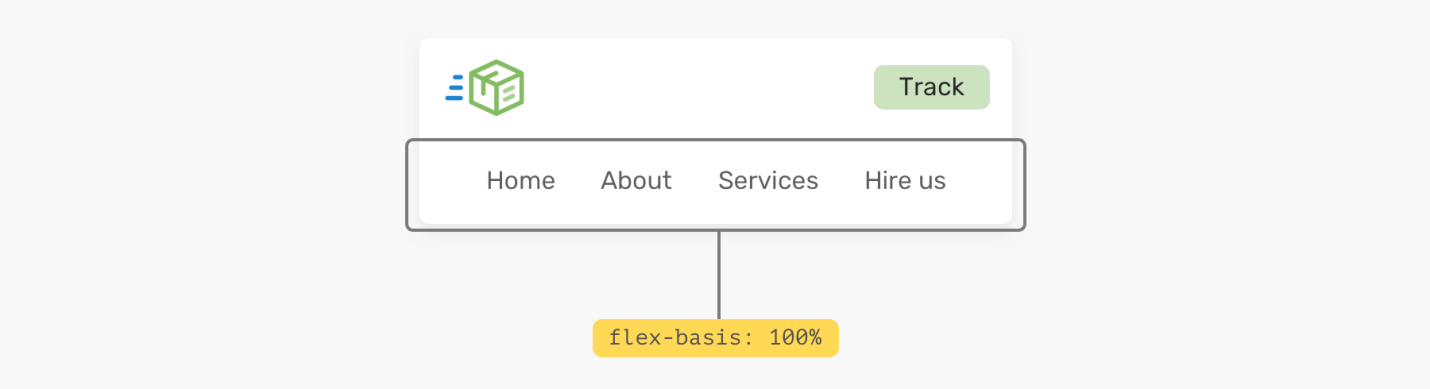


*Логотип больше не выровнен по центру*  
  
Как видите, на предыдущем рисунке логотип больше не выровнен по центру. Об этой особенности вышеописанного подхода стоит помнить для того чтобы не столкнуться с неожиданными проблемами.  
  
Теперь, когда мы исследовали разные варианты проектирования макетов заголовочных частей сайтов и поговорили о том, как создавать такие макеты, давайте рассмотрим некоторые важные идеи, которые могут нам помочь при создании заголовочных частей страниц.

Советы по разработке заголовочных частей страниц, основанных на Flexbox

▍Свойство flex-basis

Я предпочитаю пользоваться свойством flex-basis: 100% в том случае, если элемент должен иметь полную ширину на мобильных устройствах. Например, если речь идёт о важном блоке навигационных ссылок, который нельзя скрывать.



*Результат применения свойства flex-basis: 100%*  
  
Применение этого свойства, если взглянуть на предыдущий рисунок, может показаться простым делом. В реальности же это не так. Обычно заголовочная часть сайта может иметь внутренние отступы (padding), и если мы настраиваем элемент так, чтобы он занимал бы полную ширину, этого не произойдёт до тех пор, пока не окажутся перекрытыми внутренние отступы. Правда, убирать их — непрактично, так как это подействует на другие элементы макета.  
  
Вот как можно решить эту проблему:

1. Добавим flex: 1 0 100% к навигационному блоку.
2. Поменяем, если нужно, его свойство order. Иногда приходится заботиться и о других элементах, поэтому нам нужно удостовериться в том, что навигационный блок является последним элементом.
3. Настроим внешние отступы (margin) навигационного блока, воспользовавшись отрицательным значением, равным размеру внутренних отступов. Это позволит данному блоку занять полную ширину страницы.
4. Настроим внутренние отступы (padding) так, чтобы в элементе присутствовало бы немного «воздуха».
5. И, наконец, воспользуемся свойством justify-content: center для выравнивания навигационных элементов по центру (хотя это и не особенно важно).

Вот стили (в квадратных скобках даны пункты вышеприведённого списка, к которым они имеют отношение):

.nav {

  flex: 1 0 100%; */\* [1] \*/*

  order: 2; */\* [2] \*/*

  margin: 1rem -1rem -1rem -1rem; */\* [3] \*/*

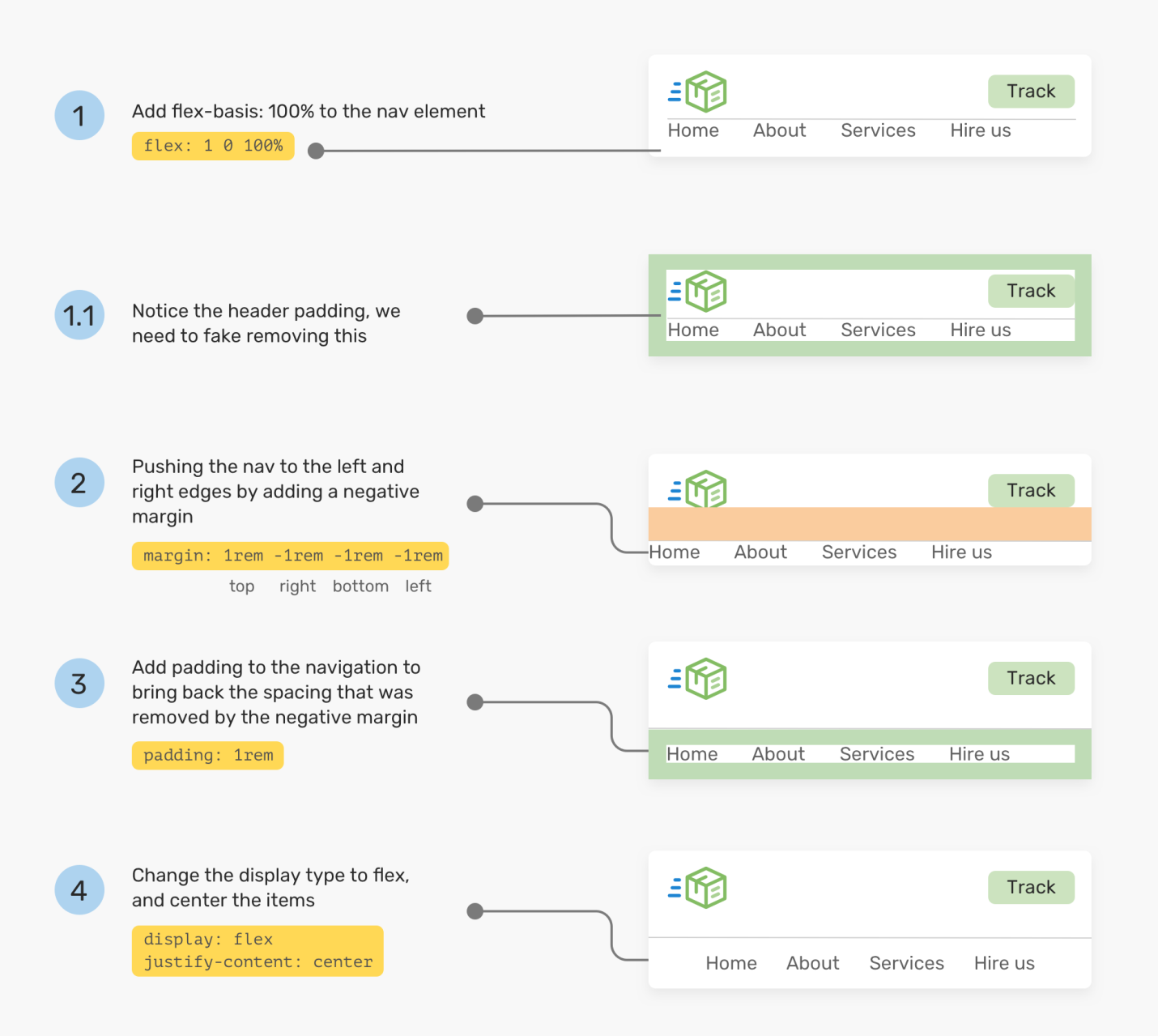
  padding: 1rem; */\* [4] \*/*

  display: flex; */\* [5] \*/*

  justify-content: center; */\* [5] \*/*

}

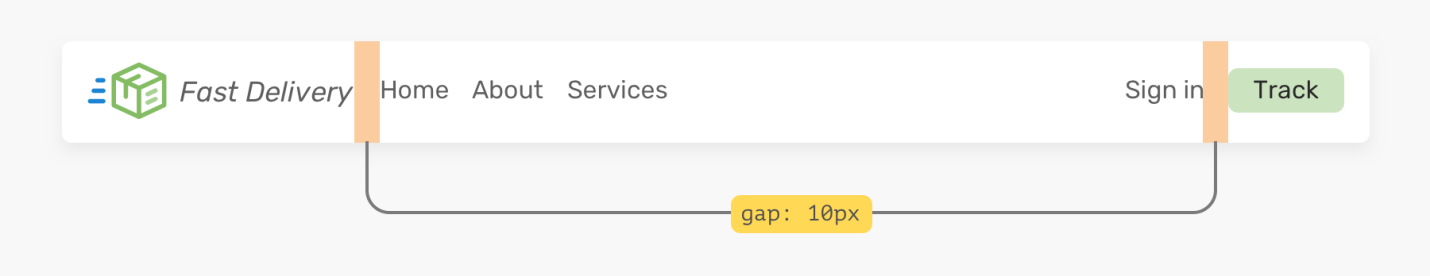
А вот как будет выглядеть пошаговое применение вышеописанных стилей к списку навигационных элементов.



*Пошаговая стилизация элементов*

▍Настройка расстояния между элементами

Теперь, когда браузеры Chrome и Firefox поддерживают свойство gap, очень легко настраивать расстояние между flex-элементами. Взгляните на следующую заголовочную часть страницы.



*Использование свойства gap*  
  
Для того чтобы настроить расстояние между элементами, выделенное на рисунке, нужно добавить к родительскому flex-элементу свойство gap: 1rem. Без использования этого свойства нам пришлось бы стилизовать элементы по-старому.

*/\* Старый подход \*/*

.brand {

  margin-right: 1rem;

}

.sign-in {

  margin-right: 1rem;

}

*/\* Новый подход \*/*

.site-header {

*/\* Другие flexbox-стили\*/*

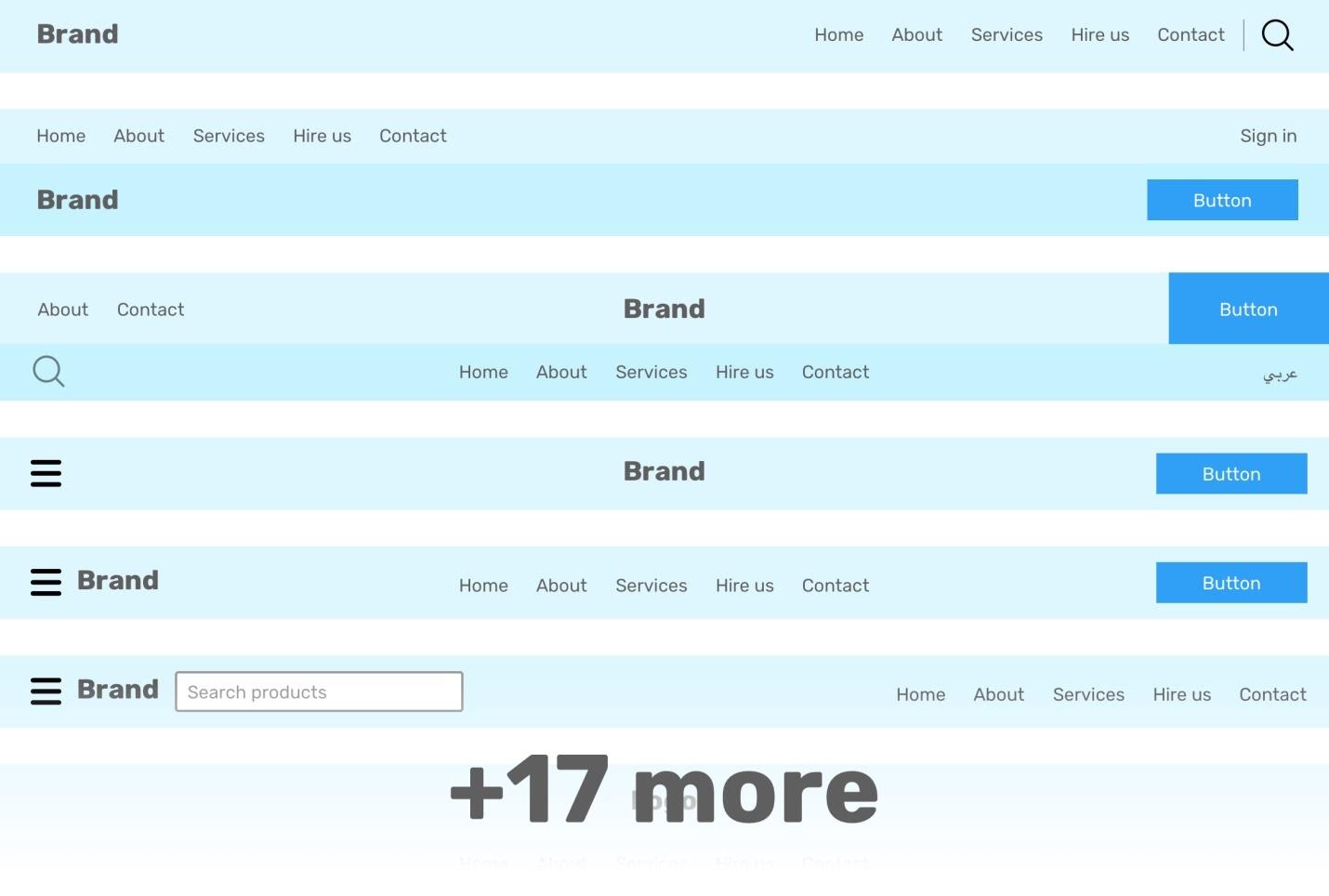
  gap: 1rem;

}

Если вы используете свойство gap — помните о том, что вам нужно будет подготовить и запасной вариант на тот случай, если оно не поддерживается браузером, в котором просматривают страницу. Я написал об этом подробную [статью](https://ishadeed.com/article/flexbox-gap/).  
  
На этом мы завершаем разговор, но прежде чем мы расстанемся, позвольте мне показать вам один мой проект.

Проект headers-css

Вот пример заголовочных частей страниц, которые созданы в рамках проекта headers-css.



*Заголовочные части страниц из headers-css*  
  
Я решил сделать проект, в котором собраны шаблоны заголовочных частей веб-сайтов, оформленные в виде отдельных страниц. Теперь, если мне понадобится подобный элемент, я смогу быстро найти его и интегрировать в новый проект. А именно, я создал 17 вариантов заголовочной части страницы. В ближайшее время я планирую расширить этот проект. Работая над этими элементами, я обращал особое внимание на следующее:

* Гибкость.
* Полностью отзывчивый дизайн.
* Использование Sass, что позволяет легко редактировать мои шаблоны (правда, в этой области мне ещё нужно немного отрефакторить код).
* Доступность (если вам попадётся что-то неправильное — пожалуйста добавьте запись в трекер задач проекта).

[Вот](https://headers-css.vercel.app/) страница, на которой можно посмотреть проект headers-css в действии. А [вот](https://github.com/shadeed/headers-css) — GitHub-репозиторий проекта.