html academy ■ Начать с нуля ▼ 😚 Получить профессию ▼ 💆 Повысить грейд ▼ Вёрстка многослойных элементов интерфейса

Выполнен на 51%

Введение

z-index

Layout

Словарь терминов

Методика вёрстки

многослойных элементов

Кейс 1, лёгкий уровень

Кейс 3, лёгкий уровень

Кейс 4, лёгкий уровень

№ Кейс 5, средний уровень

☑ Кейс 6, средний уровень

🖸 Кейс 7, средний уровень

🖸 Кейс 8, сложный уровень

Б Кейс 9, сложный уровень

🖸 Кейс 10, сложный уровень

№ Кейс 2, лёгкий уровень

Углублённая теория

Позиционирование

🖪 Как работает position: sticky

Многослойность с помощью CSS Grid

В Про vh, vw и другие единицы измерения

Главная / Моё обучение / Вёрстка многослойных элементов интерфейса / Углублённая теория /

Как работает position: sticky

По первым четырём значениям позиционирования всё интуитивно понятно и просто. Основное различие между static и relative, absolute и fixed в том, какое место они занимают в потоке документа. Элементы с позицией static и relative сохраняют своё естественное положение в потоке документа, в то время как absolute и fixed «вырываются» из потока документа и становятся плавающими.

🦝 Виталий Ризун 🔻

¥ Моё обучение ▼

Значение sticky похоже на все предыдущие значения сразу и в то же время имеет свои особенности. Разберём в деталях, как именно липкое позиционирование работает.

Оно требует определить три вещи: сам «липкий» элемент, его контейнер, который мы дальше будем звать липким контейнером, и позицию, начиная с которой элемент будет становиться липким.

«Липкий» элемент — это элемент, которому мы задали position: sticky. «Липкий» элемент может залипать

только в пределах родительского контейнера. «Липкий» контейнер — это элемент, который оборачивает «липкий» элемент. Это максимальная область, в которой может перемещаться наш элемент. Липким контейнером автоматически становится родитель липкого

элемента. Если элемент с «липким» позиционированием перестаёт залипать, это может быть связано

с недостаточными размерами родительского элемента. Элемент будет становиться «липким», как только пользователь при скролле веб-страницы достигнет определённой позиции родителя «липкого» элемента (left, top, right или bottom, отличные от значения

auto), например (top: 20px). До этого момента «липкий» элемент можно рассматривать как относительно спозиционированный. Если подытожить, для того чтобы элемент стал «липким», необходимо:

— задать для элемента position: sticky,

— размеры непосредственного родителя позволяют элементу становится «липким», — указать для элемента одно из CSS-свойств: left, top, right или bottom отличное от auto.

Разберём на примерах.

Без перечисленных трёх составляющих «липкое» позиционирование не работает.

Для начала просто сделаем элемент «липким». Представим, что у нас есть сайт, который состоит из шапки, основного содержимого и подвала. Шапку сделаем «липкой», причём она будет залипать, как только **body**

достигнет позиции top: 20px. «Липким» контейнером в этом случае будет тег body. «Липкий» контейнер

```
header
   «Липкий» элемент
                            main content
                                 footer
«Липкая» шапка на сайте
Разметка:
```

<body>

position: sticky;

top: 20px;

Стили:

<body>

top: 0;

.component {

top: 0;

страницы. Что-то вроде этого:

position: sticky;

<header>Шапка</header>

```
<main>main content</main>
   <footer>footer</footer>
 </body>
Стили:
 .header {
```

Мы хотим сделать «липким» элементом не всю шапку, а только список ссылок на разделы сайта. Но меню

«Липкий» элемент и «липкий» контейнер совпадают по высоте

плавающим — размеры «липкого» контейнера позволяют.

Рассмотрим ещё один пример, неудачный.

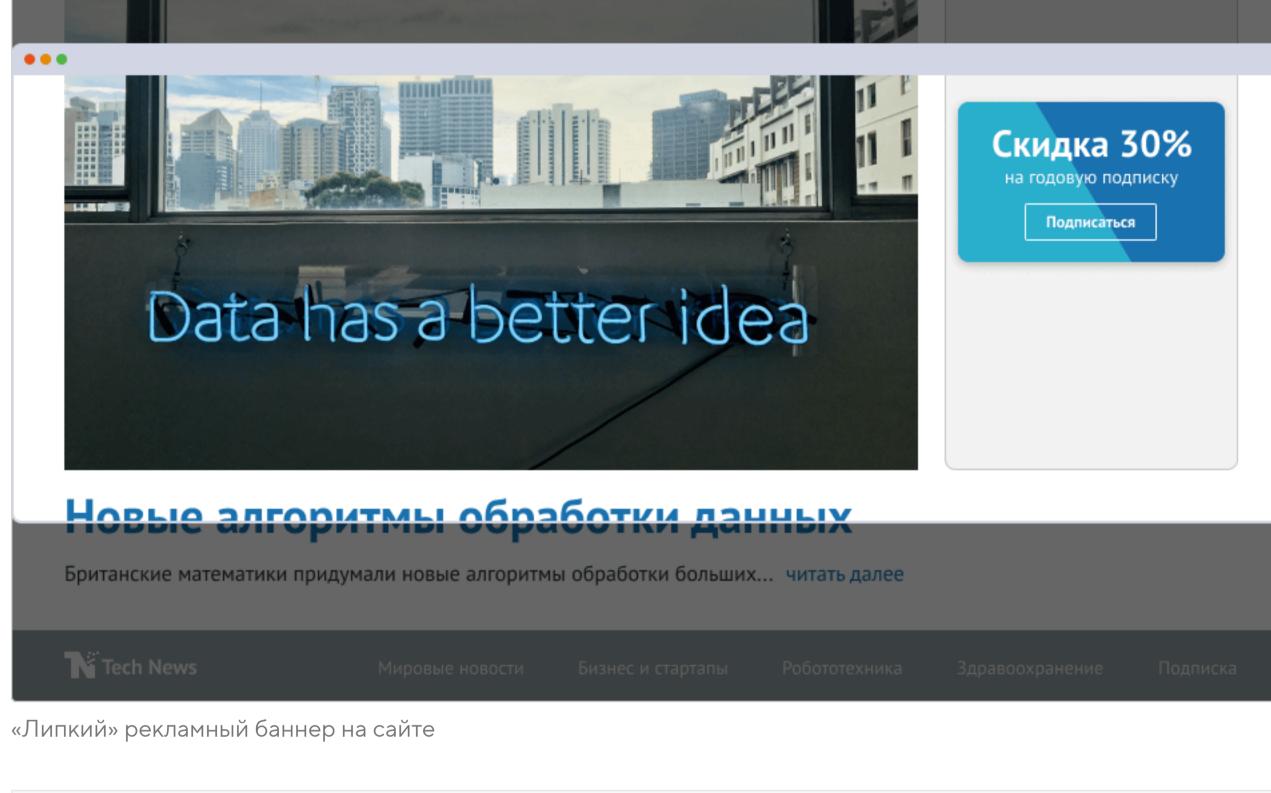
<header class="header">header

находится внутри тега nav. По высоте эти элементы примерно совпадают. В этом случае position: sticky не даст желаемого результата. При прокрутке страницы «липкий» контейнер быстро уйдёт из области видимости и уведёт за собой «липкий» элемент. Open World 🙎 Личный кабинет 🔻 🕦 О нас Контакты 🄯 Услуги Новости

```
Разметка:
<nav class="nav">
  <a href="#">
   <img src="img/logo.svg" width="120" height="50" alt="">
  </a>
  </nav>
```

```
.sticky-list {
   position: sticky;
   top: 0;
На изображении ниже «липким» элементом должен быть рекламный баннер. Серой рамкой выделен «липкий»
контейнер, отдельно показаны размеры экрана. При этой компоновке элементов баннер будет становиться
```

Tech News Бизнес и стартапы Робототехника



<main> <div class="wrapper"> <article class="main-article">Основная статья</article> <aside class="aside">

```
<div class="banner">Баннер</div>
     </aside>
    </div>
 </main>
</body>
.wrapper {
 display: flex;
.main-article {
 width: 75%;
.aside {
 width: 25%;
.banner {
 position: sticky;
```

Ho с помощью position: sticky можно прикрепить элемент и к нижнему краю страницы. К примеру, подвалу

В большинстве случаев вы будете использовать position: sticky, чтобы прикрепить элемент к верхнему краю

```
сайта можно задать «липкое» позиционирование, и при скролле он всегда будет залипать у нижнего края. И когда
мы дойдём до конца «липкого» контейнера, подвал сайта остановится на своей естественной позиции. Эту
особенность лучше использовать для элементов, которые находятся в самом конце контейнера.
Пример:
```

header main content

```
«Липкий» элемент
                                           footer
«Липкий» подвал сайта
HTML
 <main class="main-container">
   <header class="main-header">header
   <div class="main-content">main content</div>
   <footer class="main-footer">footer</footer>
 </main>
CSS
```

.main-footer {

bottom: 0;

position: sticky;

000 «НАШ РАЙОН»

>3астройщик

- Браузерная поддержка

< Позиционирование

```
В реальной жизни такое поведение можно использовать для сводных таблиц. Обычно объём данных в сводных
таблицах большой, и когда шапка таблицы уходит из поля зрения, сложно понять, что за данные отображаются
в той или иной ячейке таблицы. За счёт position: sticky можно зафиксировать шапку таблицы с названиями
колонок, это улучшит восприятие информации.
        ▼ Застройщик
                                 Юридический адрес
                                                                                       ИНН
```

000 «СТРОЙЭТАЖ» г. Владимир, улица Горького, дом 2, корпус 7, комната 1 0987654321 Сводная таблица с «липкой» горизонтальной шапкой

г. Москва, Нахимовский проспект, дом 8, строение 23

1234567890

```
Юридический адрес
  VHH
 OOO «НАШ РАЙОН»
  г Москва, Нахимовский проспект, дом 8, строение 23
  1234567890
 000 «CTPOЙЭТАЖ»
  <td>г Владимир, улица Горького, дом 2, корпус 7, комната 1</td>
  0987654321
 .developers-row-sticky {
```

```
position: sticky;
   top: 10px;
С помощью position: sticky можно фиксировать не только заголовочные строки сводных таблиц,
но и заголовочные колонки. Это может быть удобно в случае, когда таблицу нужно скролить не только сверху
вниз, но и вправо/влево. Для этого нужно обозначать, соответственно, отступ справа или слева: например, right:
10px или left: 10px для того, чтобы элемент становился липким на расстоянии в 10 пикселей от боковой
стороны своего родительского контейнера.
```

До 2020 года, чтобы использовать свойство position: sticky и быть уверенным, что оно сработает приходилось подключать **полифилы** (полифиллы;англ. Polyfill). Самый популярный из них Stickyfill. В настоящее время Stickyfill не поддерживается и в последнем своём коммите авторы написали: Unmaintained! Stickyfill did a good job while the browsers were implementing position: sticky support. You can now safely use stickies without a polyfill, all modern browsers support them natively.

современные браузеры его поддерживают из коробки. У свойства и на самом деле хорошая браузерная поддержка, в чём можно убедиться на CanlUse.

поддержку position: sticky. Теперь вы можете использовать position: sticky без полифилла, все

Перевод: Необслуживаемый! **Stickyfill** проделал хорошую работу, пока браузеры реализовывали

Тренажёры Подписка Для команд и компаний Учебник по РНР Профессии Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Работа наставником

Услуги

Для учителей

Стать автором

Практикум

TypeScript. Теория типов

Webpack

Git и GitHub

© ООО «Интерактивные обучающие технологии», 2013–2023

Курсы

JavaScript. Архитектура клиентских приложений React. Разработка сложных клиентских приложений Node.js. Профессиональная разработка REST API Node.js и Nest.js. Микросервисная архитектура Алгоритмы и структуры данных Паттерны проектирования

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

Блог

С чего начать

Отчеты о курсах

Информация

О центре карьеры

Об Академии

Остальное

Написать нам

Мероприятия

Форум

Шпаргалки для разработчиков

z-index >

Сведения об образовательной организации Конфиденциальность

Анимация для фронтендеров

Vue.js 3. Разработка клиентских приложений