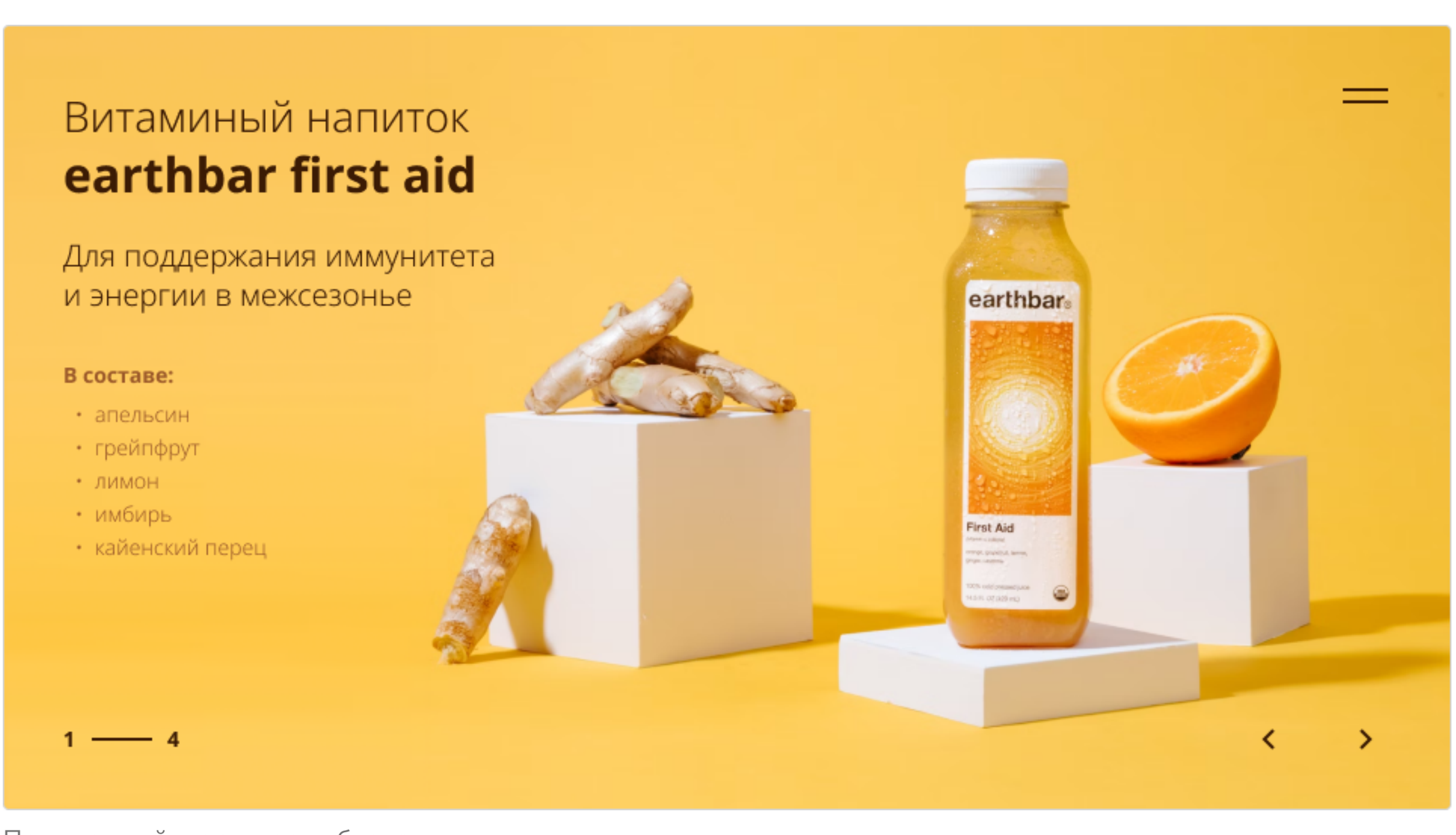
Вёрстка компонента Карусель/Слайдер

Ещё один компонент, который можно отнести к многослойным, — это компонент для показа слайдов. Он состоит из набора слайдов и элементов управления ими. Это могут быть кнопки назад и вперёд, кнопки перехода к нужному слайду. Иногда используется индикатор с номером текущего слайда и общим их количеством. Индикатор может быть где угодно: снизу слева, снизу справа или посередине. На отдельном слайде размещается либо картинка, либо полноценный раздел с заголовком, текстовым описанием, изображением и ссылкой для перехода на соответствующую страницу сайта. Как правило, слайдеры используют в промоблоке и галерее фотографий.

В библиотеках компонентов его обычно называют *Carousel*. Другое его название *Slideshow* или *Slider*.



Пример слайдера в промоблоке

В нашем примере, помимо списка слайдов, в слайдере есть кнопки назад и вперёд и индикатор текущего слайда (*1 из 4*). Напишем примерную разметку компонента. Отдельный слайд обернём в тег `<section>`.

```
<div class="slider">
  <!-- Список слайдов -->
  <ul class="slider-list">
    <li class="slider-list-item" id="slide-1">
      <!-- Отдельный слайд -->
      <section class="slide">
        <h2 class="slide-title">Витаминный напиток <span>earthbar first aid</span></h2>
        <p class="slide-description">Для поддержания иммунитета и энергии в межсезонье</p>
        <h3 class="slide-list-title">В составе:</h3>
        <ul class="slide-list">
          <li>апельсин</li>
          <li>грейпфрут</li>
          <li>лимон</li>
          <li>имбирь</li>
          <li>кайенский перец</li>
        </ul>
      </section>
    </li>
    <!-- Другие слайды -->
    ...
  </ul>
  <!-- Кнопки назад/вперёд -->
  <button class="slider-button slider-button-prev" disabled>Назад</button>
  <button class="slider-button slider-button-next">Далее</button>
  <!-- Индикатор текущего слайда -->
  <div class="slider-counter">
    <span class="slider-counter-count">1</span>
    <span class="slider-counter-delimiter"></span>
    <span class="slider-counter-count">4</span>
  </div>
</div>
```

Разберём поведение слайдера, элементов внутри него и накидаем примерные стили.

Все процессы выполняются внутри области просмотра: смена слайдов, положение элементов управления. Определим размеры области просмотра и скроем всё, что находится за её пределами `overflow: hidden`.

```
.slider {
  width: 960px;
  height: 520px;
  overflow: hidden;
}
```

А дальше нам понадобится добавить многослойность. Список слайдов займёт всю область просмотра. Растянем его по всей ширине и высоте. Кнопкам «Назад», «Вперёд» и индикатору установим абсолютное позиционирование (`position: absolute;`) и укажем координаты за счёт свойств `left`, `right`, `bottom`. Точкой отсчёта будет область просмотра, элемент с классом `slider`, для него добавим `position: relative;`.

```
.slider {
  position: relative;
  width: 960px;
  height: 520px;
  overflow: hidden;
}

.slider-list {
  width: 100%;
  height: 100%;
  padding: 0;
  margin: 0;
  list-style: none;
}

.slider-button {
  position: absolute;
  bottom: 20px;
  width: 54px;
  height: 54px;
}

.slider-button-prev {
  right: 95px;
}

.slider-button-next {
  right: 30px;
}

.slider-counter {
  position: absolute;
  left: 40px;
  bottom: 40px;
}
```

Замечание

Для кнопки с атрибутом `disabled` можно указать `pointer-events: none;`, так она не будет реагировать на наведение курсора (*нажатие, перетаскивание и иже с ними*). Подробнее о CSS-свойстве читайте в спецификации. Браузерную поддержку CSS-свойства можно посмотреть на CanIUse.

Слайды внутри списка выстроены в ряд. Добиться этого поведения можно за счёт *CSS Flex Layout*. Установим для списка `display: flex;`. По умолчанию элементы списка будут сжиматься и стараться уместиться в окне просмотра. Отменить это поведение можно установив `flex-shrink: 0`, либо `flex: none;`, что будет равнозначно `flex: 0 0 auto;`.

```
.slider-list {
  display: flex;
  width: 100%;
  height: 100%;
  padding: 0;
  margin: 0;
  list-style: none;
}

.slider-list-item {
  flex: none; /* Не даём отдельному слайду сжаться */
  width: 100%;       /* Растягиваем его по всей ширине контейнера */
}
```

Передвижение слайдов внутри области просмотра обычно реализуют с помощью `transform: translate`. Чтобы показать второй слайд, для списка нужно указать смещение `transform: translate3d(-960px, 0, 0);`. Чтобы показать третий — `transform: translate3d(-1920px, 0, 0);`. Напомним, что `960px` — это ширина одного слайда.

```
...
<ul class="slider-list" style="transform: translate3d(-960px, 0, 0);">
...
</ul>
...
```

```
...
<ul class="slider-list" style="transform: translate3d(-1920px, 0, 0);">
...
</ul>
...
```

Автоматизируют процесс с помощью *JavaScript*.

Можно ограничиться использованием `transform: translateX()`. Вариант `transform: translate3d()` более универсальный и нужен для того, чтобы вынести трансформации на отдельный слой вычислений.

Как только мы добавляем CSS-свойство `transform` в правило, появляется возможность задать плавный переход между слайдами.

```
.slider-list {
  display: flex;
  width: 100%;
  height: 100%;
  padding: 0;
  margin: 0;
  list-style: none;
  transition: transform 300ms ease;
}
```

− Немного про доступность

Слайды в слайдере у нас обёрнуты в тег `<ul>`, и программа для чтения с экрана (скринридер) считает их как список. Она сообщит что-то вроде «список, 4 элемента». Примечательно, что в *Safari*, когда вы устанавливаете `list-style: none`, это не работает. Команда WebKit решила удалять семантику списка, когда список не похож на список. Но есть возможность перехитрить систему — добавить `role="list"` для списка. Например, так:

```
<ul role="list">
  <li>...</li>
  ...
</ul>
```

Подробнее читайте в статье Огласите количество предметов.

Есть много примеров компонентов, где слайдер реализован на *чистом HTML+CSS* без использования *JavaScript*.

В этом случае используется встроенная возможность браузера переходить к якорному элементу. Часто в таких вариантах реализации используются CSS-свойства `scroll-snap-type`, `scroll-behavior` и `scroll-snap-align` для плавного прокручивания слайдов.

Примечательно, что для каждого слайда создаются свои элементы управления — кнопки «Назад», «Вперёд». Приведём примерную разметку и стили для этого способа реализации.

```
<div class="carousel">
  <div class="carousel-item slide" id="slide1">
    <img class="slide-img" src="#" alt="Изображение для первого слайда">
    <div class="slide-buttons">
      <a class="button button-circle" href="#slide3">‹</a>
      <a class="button button-circle" href="#slide2">›</a>
    </div>
  </div>
  <div class="carousel-item slide" id="slide2">
    <img class="slide-img" src="#" alt="Изображение для второго слайда">
    <div class="slide-buttons">
      <a href="#slide1" class="button button-circle">‹</a>
      <a href="#slide3" class="button button-circle">›</a>
    </div>
  </div>
  <div class="carousel-item slide" id="slide3">
    <img class="slide-img" src="#" alt="Изображение для третьего слайда">
    <div class="slide-buttons">
      <a class="button button-circle" href="#slide2">‹</a>
      <a class="button button-circle" href="#slide1">›</a>
    </div>
  </div>
</div>
```

```
.carousel {
  display: flex; /* Выстраиваем слайды в ряд */
  overflow-x: scroll; /* Добавляем горизонтальный скролл */
  scroll-snap-type: x mandatory; /* Контейнер привязан только по своей горизонтальной оси (слайдер горизонтальный) */
  scroll-behavior: smooth; /* Контейнер плавно прокручивается */
}

.carousel-item {
  position: relative;
  box-sizing: content-box;
  display: flex;
  flex: none; /* Не даём слайдам сжаться, они растянуты по ширине содержимого, то есть 100% ширины контейнера */
  scroll-snap-align: start; /* Левый край элемента должна прижиматься к контейнеру (слайдер горизонтальный) */
}

.slide-img {
  width: 100%;
  max-width: 100%;
  height: auto;
}

.slide-buttons {
  position: absolute;  /* Растягиваем элементы управления по всей ширине контейнера */
  left: 0;
  right: 0;
  top: 50%;
  transform: translateY(-50%);
  display: flex;
  justify-content: space-between;
}
```

− Про значения CSS-свойства scroll-snap-type

Значение `mandatory` определяет поведение, при котором всякий раз, когда пользователь прекращает прокрутку, браузер должен вернуться к точке привязки. Другое значение этого CSS-свойства — `proximity`, менее строгое. Оно означает, что браузер может возвращаться к точке привязки, если ему это покажется уместным. Значение `mandatory` делает работу интерфейса более последовательной и предсказуемой, но, как говорится в спецификации, оно также может быть достаточно опасным. Представьте ситуацию: внутри прокручиваемого контейнера расположен элемент, высота которого превышает область видимости. Если для контейнера задано `scroll-snap-type: mandatory`, он всегда будет привязан либо кверху этого элемента, либо кверху элемента, расположенного ниже и зафиксировать прокрутку в середине высокого элемента уже невозможно.

Вместо смещения «влево или вправо», на слайдере можно использовать эффект затухания (*Fade effect*). Чтобы эффект заработал, слайды надо сложить в стопку.

Разберём, какие стили для этого нам понадобятся.

Отдельным слайдам устанавливаем `position: absolute`. Скрываем их с помощью CSS-свойств `opacity` и `visibility`. Это нужно, чтобы подключить анимацию проявления.

```
.carousel {
  position: relative;
  width: 480px;
  height: 550px;
  overflow: hidden;
}

.screens {
  position: relative;
  margin: 0;
  padding: 0;
  list-style: none;
}

.screen {
  position: absolute;
  top: 0;
  left: 0;
  opacity: 0;
  visibility: hidden;
  transition: opacity 0.6s ease;
}

.screen.active {
  opacity: 1;
  visibility: visible;
}

...
```

В таком варианте слайды будут наслаиваться друг на друга — неактивные становятся невидимыми, а активный, наоборот, видимым. Совместно с плавными переходами эта реализация выглядит необычно и интересно. Способ хорошо подходит для слайдера с фотографиями.

‹ Вёрстка пользовательской подсказки
Вёрстка компонента Выпадающий список ›