Документация по проекту Fashionee.

Содержание

[Введение 2](#_Toc193478073)

[1. Общие компоненты (common components) 3](#_Toc193478074)

[1.1 Header (шапка страницы) 3](#_Toc193478075)

[1.2 Content-Block (блок с контентом) 3](#_Toc193478076)

[1.3 Footer («подвал» страницы) 3](#_Toc193478077)

[2. UI компоненты (ui) 3](#_Toc193478078)

[2.1 NavLink (навиагционная ссылка) 3](#_Toc193478079)

[2.2 Socials (блок с ссылками на социальные сети) 4](#_Toc193478080)

[3. Showcase (основное содержимое страницы Shop) 4](#_Toc193478081)

[3.1 SideBar (боковое меню) 4](#_Toc193478082)

[3.2 Products (основной блок страницы) 4](#_Toc193478083)

# Введение

# Общие компоненты (common components)

## Header (шапка страницы)

Компонент Header является шапкой страницы. В нем содержатся такие элементы как:

* Бургер – меню;
* Логотип сайта;
* Навигация между страницами сайта (текстовое меню). Для него используется UI компонент NavLink;
* Навигация между страницами профиля (меню с иконками). Для него создан компонент внутри Header с названием HeaderIconMenu;

## Content-Block (блок с контентом)

Компонент Content-Block представляет собой блок с контентом, содержащим в себе:

* название страницы, на которой находится пользователь;
* две навигационные ссылки для переключения между страницами. Используется UI компонент NavLink;
* баннер с изображением.

## Footer («подвал» страницы)

Компонент Footer представляет собой блок с информацией необходимой пользователю, который размещается внизу страницы. В нем содержатся такие элементы как:

* Логотип и описание компании;
* Ссылки на социальные сети компании. Используется компонент Socials;
* Блок About, содержащий ссылки на информацию о компании и продукции. Используется компонент NavLink;
* Блок Useful Links, содержащий ссылки на информацию о поддержке, правилах использования и т.д. продукции. Используется компонент NavLink;
* Уведомление об авторском праве;
* Возможные способы оплаты;

# UI компоненты (ui)

## NavLink (навиагционная ссылка)

NavLink – «глупый», статичный компонент, который получает необходимые данные и отображает их на странице. Данный компонент получает такую информацию как:

* className – имя класса для стилизации;
* href – ссылка на страницу;
* text – текст для отображения ссылки на странице;
* img – иконка для навигационной ссылки (если таковая имеется)
  1. Socials (блок с ссылками на социальные сети)

Socials – так же статичный компонент, в который вынесено оформление блока с ссылками на социальные сети для того, что бы не перегружать визуально версткой компонент Footer.

# Showcase (основное содержимое страницы Shop)

Showcase – компонент, в котором находится содержимое страницы Shop, разделенное в свою очередь еще на 2 компонента:

* SideBar – боковое меню с поиском, фильтрами и т.д.;
* Products – основной блок с карточками товара, сортировкой и пагинацией.

## SideBar (боковое меню)

SideBar – один из компонентов страницы Shop. Содержит в себе следующие компоненты:

* Search – поиск товара;
* Filters – различные фильтры для поиска товара;
* LastViewed – последние просмотренные товары;

### Filters

Компонент Filters представляет собой общий сборный компонент для трех разных компонентов фильтров:

* CategoryFilter (женские, мужские и тд; список)
* PriceFilter (от минимального к максимальному; два инпута)
* ColorFilter (по цвету; чекбоксы)

В Filters вызывается функция getFilters, собирающая значения для фильтров, результат ее работы передается в каждый отдельный фильтр.

Компоненты фильтров с помощью infoForFilters переданным им, отрисовывают списки и значения фильтров.

## Products (основной блок страницы)

Products – основной блок страницы Shop. Содержит в себе следующие компоненты:

* Sort – компонент с сортировкой товара;
* Product – компонент карточки товара;
* Pagination – переключение между страницами;

# Helpers (вспомогательные функции)

## getFilters

getFilters – функция, которая выбирает из продукта необходимые значения для фильтров.

Что в этой функции происходит?

Для начала создается объект с начальными значениями для фильтров:

const filtersInfo = {

    categories: new Set(),

    prices: {

      min: 0,

      max: 0

    },

    colors: new Set()

  };

Для ключей categories и colors создается множество уникальных значений Set. Для ключа prices создается вложенный объект с min и max значениями цены. Изначально они равны нулю.

Далее мы приступаем к заполнению filtersInfo необходимыми значениями:

 products.forEach((product) => {

Для этого мы проходим по каждому продукту в массиве. И с каждым продуктом делаем следующее:

product.categories.forEach((category) => {

      filtersInfo.categories.add(category);

    });

Здесь мы в каждом товаре перебираем значение categories (т. к. их может быть несколько) и каждое значение записываем в Set.

Далее определяем максимальную цену:

if (product.price > filtersInfo.prices.max) {

      //если значение цены у товара больше, чем стоит в объекте, то записываем его как max

      filtersInfo.prices.max = product.price

    };

Здесь на каждой итерации происходит сравнение цены текущего продукта с уже записанным в filtersInfo. На первой итерации происходит сравнение начального значения 0 со значением цены первого продукта. Так как оно в любом случае больше 0, то в prices.max записывается цена первого продукта. На второй итерации уже сравнивается prices.max = цена первого продукта и цена второго. Таким образом, находится максимальная цена.

Далее мы заполянем Set у colors:

 filtersInfo.colors.add(product.color);

  });

Здесь все просто, так как у продукта один цвет. На этом цикл foreach заканчивается.

Далее мы преобразуем множества Set в массивы для того, чтобы в компонентах фильтров можно было работать с их значениями:

  filtersInfo.categories = [...filtersInfo.categories];

  filtersInfo.colors = [...filtersInfo.colors];

  return filtersInfo

}