2 вида отображения контента на странице

2 вида фильтров

взято отсюда <https://practicum.yandex.ru/trainer/web/lesson/80eb4f5f-55ce-4e2e-a4df-7fd4e64b0418/task/28f54a60-a440-4c74-96d2-135f2eed5702/>

**родительский базовый класс Card:**

единолично не используется, класс создан как «родитель» для DefaultCard и HorizontalCard. Здесь прописаны общие для этих классов свойства и методы, которые наследуются для указанных классов и там кастомизируются (подгоняются/дополняются).

**свойства**:

* селектор карточки (див)

**методы**:

* \_getTemplate() - создание клона шаблона, путем поиска по названию селектора
* \_handleOpenPopup() - открытие попапа карточки (переменная ссылки картинки и класс попапа берется импортом и меняются их значения)
* \_handleClosePopup() - аналогично закрытию
* \_setEventListeners() - установка слушателя на клик по товару, чтобы открылся попап + слушатель на кнопку закрыть

**класс DefaultCard на основе Card:**

отвечает за генерацию html кода карточки в стандартном формате, без цены и описания. Есть дополняющие методы открытия/закрытия попапа, которые дополняют такой же метод у класса Card, дополняя инфой с описанием

**свойства**:

* селектор карточки (див) - наследование от Card
* this.\_title - заголовок 1 карточки на основе данных из массива
* this.\_description - то же самое описание
* this.\_image - также картинка

**методы**:

* generateCard() - на основе родительского \_getTemplate() клонирует шаблон, устанавливает родительского слушателя \_setEventListeners(), подменяет картинку, описание и заголовок, а потом возвращает всю карточку
* \_handleOpenPopup() - на основе родительского \_handleOpenPopup(), но с опцией изменения описания карточки у попапа
* \_handleClosePopup() - на основе родительского \_handleClosePopup(), но с опцией удаления описания у попапа

**класс HorizontalCard на основе Card:**

отвечает за генерацию html кода карточки в расширенном, горизонтальном формате, где еще добавляется цена и подробное описание. У класса нет дополнительных методов открытия/закрытия попапа, т.к. при таком просмотре попап отображается без цены (как в Card)

**свойства**:

* селектор карточки (див) - наследование от Card
* this.\_title - заголовок 1 карточки на основе данных из массива
* this.\_description - то же самое описание
* this.\_image - также картинка
* this.\_price - также цена

**методы**:

generateCard() - на основе родительского \_getTemplate() клонирует шаблон, устанавливает родительского слушателя \_setEventListeners(), подменяет картинку, описание, цену и заголовок, а потом возвращает всю карточку

**класс Section:**

занимается выводом на странице переданного ему контента. Сам контент он не генерирует, а лишь обрабатывает переданный ему. Для каждого экземпляра вывода может быть своя функция обработки элементов. Например, для карточек товара вывод на основе классов DefaultCard или HorizontalCard, а для вывода кнопок фильтрации – на основе класса FilterButton.

Внутри экземпляра есть функция renderer, в котором создаются отдельные экземпляры карточек товара или кнопок согласно полученным данным из массива.

**свойства**:

* this.\_renderedItems - массив items, это список товаров или список кнопок, то есть список того, что отобразить
* this.\_renderer - колбек функция обработки каждого элемента из списка. Тут самой функции нет, только переменная, функция будет написана в момент присвоения экземпляра. Делается, чтобы у классов была "слабая связь", то есть не прописаны конкретные условия использования других классов
* this.\_container - селектор контейнера, куда будет вставляться контент

**методы**:

* clear() - очистка контента в контейнере дива перед добавлением нового контента. Очищение нужно, чтобы не было дополнения в конец, а полная замена контента
* renderItems() - рендерит: сначала очищает контент с помощью clear(), потом проходит по каждому элементу массива, и внутри цикла forEach использует колбек-функцию this.\_renderer, которая пишется в момент присвоения экземпляра
* setItem(element) - берет элемент и вставляет в html верстку, в указанный в конструкторе див (this.\_container)

**класс FilterButton**:

отвечает за отображение (генерацию кода) кнопок, при нажатии на которые происходит смена формата отображения карточек (например DefaultCard – карточки без цен и описания или HorizontalCard с описанием и ценой)

**свойства**:

* this.\_buttonSelector - селектор кнопки, которая в шаблоне для последующего копирования
* this.\_additionalButtonClass = data.buttonClass;
* this.\_handleButtonClick = handleButtonClick – колбек функция, которая будет создаваться в момент создания нового экземпляра (описание процесса отображения кнопок фильтрации)
* this.\_isGrid = data.isGrid – это способ отображения, true – default, а false это горизонтальный способ отображения

**методы**:

* \_getTemplate() - создание клона шаблона, путем поиска по названию селектора. Приватный метод
* generateButton() - на основе getTemplate() клонирует шаблон, устанавливает слушателя \_setEventListeners(),а потом возвращает html код кнопки
* \_setEventListeners() – установка слушателя на клик по кнопке. При нажатии будет выполняться колбек функция handleButtonClick, которая будет написана в момент создания экземпляра

**Файл index.js**

Связь разных классов, создание экземпляров классов происходит в этом файле. Каждый класс должен работать независимо, а способы их связи д.б. написаны в колбек функциях во время создания экземпляров.

Что происходит:

* Импорт классов из соответствующих файлов
* Импорт констант из /utils/constants.js
* Отображение карточек товара в базовом виде (DefaultCard):
* создание экземпляра defaultCardList класса Section, которому передается массив товаров-карточек + функция renderer. Section внутри функции renderer обрабатывает каждый элемент массива
* Renderer(): Для каждого элемента массива создается свой экземпляр класса DefaultCard – то есть своя карточка товара со своими слушателями, кодом html. В случае с примером из тренажера, это будет 3 таких карточки.
* Каждая полученная на предыдущем этапе карточка товара подготовлена для добавления на страницу методом setItem(), но это произойдет только после вызова метода renderItems() для экземпляра defaultCardList.
* В самом конце кода вызывается написанный экземпляр defaultCardList и его метод renderItems(). Отображение карточек на странице произойдет только после такого вызова метода. До этого все карточки товаров и их перенос на страницу только был написан, но не исполнен. Это нужно, чтобы при первой загрузке страницы отображались карточки в обычном виде.
* Отображение карточек товара в расширенном виде (HorizontalCard):
* создание экземпляра horizontalCardList класса Section, которому передается массив товаров-карточек + функция renderer. Section внутри функции renderer обрабатывает каждый элемент массива.
* Renderer(): Для каждого элемента массива создается свой экземпляр класса HorizontalCard – то есть своя карточка товара со своими слушателями, кодом html. В случае с примером из тренажера, то это будет 3 таких карточки.
* Каждая полученная на предыдущем этапе карточка товара подготовлена для добавления на страницу методом setItem(), но это произойдет только после вызова метода renderItems().
* Написанный экземпляр horizontalCardList сразу не вызывается и не исполняется. Он будет вызван при нажатии на соответствующую кнопку фильтрации на этапе 5.
* Отображение кнопок фильтрации, которые будут менять способ отображения карточек (DefaultCard, HorizontalCard):
* создание экземпляра filterList класса Section, которому передается массив из 2 видов отображения кнопки + функция renderer. Section внутри функции renderer обрабатывает каждый элемент массива.
* Renderer(): Для каждого элемента массива создается свой экземпляр класса кнопки фильтрации. В примере 2 таких кнопки.
* Каждая полученная на предыдущем этапе кнопка подготовлена для добавления на страницу методом generateButton() класса кнопок
* Внутри каждого экземпляра кнопки также есть слушатель нажатия кнопок setEventListeners(). В случае нажатия кнопки вызывается метод handleButtonClick(), которому можно передать аргумент isGrid. В изначальном массиве кнопок из п.1. этот параметр записан для каждого вида кнопок (в формате true или false), и эта опция нам служит для их отличия между собой.
* если у карточки есть переменная isGrid, то нажатие на первую кнопку вызовет метод defaultCardList.renderItems(), если нет, то будет рендериться horizontalCardList.
* В самом конце кода вызывается написанный экземпляр filterList класса Section и его метод renderItems(). Отображение кнопок фильтрации на странице произойдет только после вызова этого метода. До этого все карточки товаров и их перенос на страницу был только написан, но не исполнен. Это нужно, чтобы при первой загрузке страницы отображались кнопки.