

Проектная работа "Веб-ларек"

Стек: HTML, SCSS, TS, Webpack

Структура проекта:

- `src/` — исходные файлы проекта
- `src/components/` — папка с JS компонентами
- `src/components/base/` — папка с базовым кодом

Важные файлы:

- `src/pages/index.html` — HTML-файл главной страницы
- `src/types/index.ts` — файл с типами
- `src/index.ts` — точка входа приложения
- `src/styles/styles.scss` — корневой файл стилей
- `src/utils/constants.ts` — файл с константами
- `src/utils/utils.ts` — файл с утилитами

Установка и запуск

Для установки и запуска проекта необходимо выполнить команды

```
npm install
npm run start
```

или

```
yarn
yarn start
```

Сборка

```
npm run build
```

или

```
yarn build
```

Архитектура

Базовый код

Проект представляет реализацию архитектуры MVP(Model-View-Presenter) где:

Слой MODEL - отвечает за работу с данными. В него входят классы Api выполняют роль интерфейса для взаимодействия с удаленным сервером или API, предоставляя методы для выполнения запросов и обработки ответов.

Класс Api

предоставляет методы для выполнения HTTP-запросов к удаленному серверу.

Конструктор:

constructor(baseUrl: string, options: RequestInit = {}): Создает новый экземпляр класса Api.

Методы:

get(uri: string): Promise: Выполняет HTTP GET запрос по указанному URI.

post(uri: string, data: object, method: ApiPostMethods = 'POST'): Promise: Выполняет HTTP POST, PUT или DELETE запрос по указанному URI.

Защищенные методы:

handleResponse(response: Response): Promise: Обрабатывает ответ от сервера и разрешает или отклоняет промис в зависимости от статуса ответа.

Типы данных:

ApiResponse: Объект ответа от сервера, содержащий общее количество элементов и массив элементов указанного типа.

ApiPostMethods: Тип, представляющий поддерживаемые методы запроса ('POST', 'PUT' или 'DELETE').

Класс: AuctionAPI Расширяет функциональность базового класса Api.

Конструктор:

constructor(cdn: string, baseUrl: string, options?: RequestInit): Создает новый экземпляр класса AuctionAPI.

Методы:

getLotList(): Promise<Item[]>: Получает список лотов с сервера.

orderLots(order: IOOrder): Promise: Отправляет заказ на сервер.

Типы данных

Item: Структура данных лота.

IOOrder: Структура данных заказа.

IOOrderResult: Структура данных результата заказа.

Слой PRESENTER - отвечает за управление бизнес-логикой и взаимодействием между моделью (Model) и представлением (View).

Абстрактный класс Model-

Этот абстрактный класс предоставляет базовый функционал для моделей

Конструктор: Принимает частично заполненную модель (Partial) и объект событий (events: IEvents).

Копирует свойства из данных в текущий объект.

Метод emitChanges(event: string, payload?: object): Отправляет событие event с данными payload через объект событий.

Этот класс служит основой для создания конкретных моделей данных в приложении, обеспечивая им средства для управления своим состоянием и уведомления о своих изменениях

Класс EventEmitter -

Реализует паттерн «Наблюдатель» и позволяет подписываться на события и уведомлять подписчиков

о наступлении события.

Класс имеет методы on , off , emit — для подписки на событие, отписки от события и уведомления подписчиков о наступлении события соответственно

Класс AppState Расширяет функциональность абстрактного базового класса Model -

включает в себя методы для обработки данных, валидации, управления состоянием приложения и выполнения действий в ответ на действия пользователя. Эти методы обрабатывают данные модели и принимают решения о том, какие действия следует выполнить в ответ на изменения состояния.

Методы:

- setCatalog(items: IItem[]): void
Устанавливает каталог товаров приложения.
Принимает массив элементов каталога и создает экземпляры LotItem.
Излучает событие об изменении каталога.
- addItemInBasket(items: LotItem): void
Добавляет товар в корзину покупок.
- getIdOrder(): string[]
Возвращает массив идентификаторов товаров в корзине покупок.
- deleteItemOrder(item: LotItem): void
Удаляет товар из корзины покупок.
- clearBasket(): void
Очищает корзину покупок.

- `getTotal(): number`
Возвращает общую стоимость товаров в корзине.
- `checkAddBasket(item: LotItem): boolean`
Проверяет, добавлен ли товар в корзину покупок.
- `setOrderFieldPay(field: keyof IOrderForm, value: string): void`
Устанавливает значение поля заказа оплаты и выполняет его валидацию.
- `setOrderField(field: keyof IOrderForm, value: string): void`
Устанавливает значение поля заказа и выполняет его валидацию.
- `checkPay(data: PayButtons): string`
Проверяет состояние кнопки оплаты и обновляет его при необходимости.
- `validateOrderPay(): boolean`
Выполняет валидацию данных заказа оплаты и возвращает результат валидации.
- `validateOrder(): boolean`
Выполняет валидацию данных заказа и возвращает результат валидации.

Класс LotItem Расширяет функциональность абстрактного базового класса Model -

представляет отдельный товар в каталоге. Он содержит информацию о товаре и его текущем статусе в корзине. является важной частью структуры данных приложения, предоставляя информацию о каждом товаре и обеспечивая функциональность для управления ими

Слой VIEW - отвечает за отображение данных пользователю и за реагирование на действия пользователя

Абстрактный класс Component - предоставляет базовый функционал для работы с компонентами пользовательского интерфейса.

Конструктор: Принимает корневой HTML-элемент контейнера, в котором будет размещен компонент. Выполняет общие инициализации.

Методы

- `toggleClass(element: HTMLElement, className: string, force?: boolean)`
Переключает класс className у элемента element.
- `addClass(element: HTMLElement, className: string)`
Добавляет класс className элементу element.
- `removeClass(element: HTMLElement, className: string)`
Удаляет класс className у элемента element.
- `setText(element: HTMLElement, value: unknown)`
Устанавливает текстовое содержимое элемента element.

- `setDisabled(element: HTMLElement, state: boolean)`
Устанавливает состояние блокировки элемента `element` в зависимости от `state`.
- `setHidden(element: HTMLElement)`
Скрывает элемент `element`.
- `setVisible(element: HTMLElement)`
Отображает элемент `element`.
- `setImage(element: HTMLImageElement, src: string, alt?: string)`
Устанавливает изображение с альтернативным текстом для элемента `element`.
- `render(data?: Partial): HTMLElement`
Метод для отображения компонента. Принимает данные с необязательными свойствами и возвращает корневой HTML-элемент компонента.

Класс Card - Расширяет функциональность абстрактного базового класса Component

Карточка товара для главной страницы Этот класс создает карточку товара на основе указанного HTML-шаблона. Он предназначен для использования на главной странице, где карточки товаров представляются в большом формате.

Конструктор: Принимает название блока (CSS-класс), контейнер (HTML-элемент) и действия (колбэк) для кнопки. Инициализирует свойства класса, представляющие элементы карточки.

Методы:

- `setId(value: string)`
Устанавливает идентификатор для карточки.
- `getId(): string`
Возвращает идентификатор карточки.
- `setTitle(value: string)`
Устанавливает название товара.
- `setPrice(value: number)`
Устанавливает цену товара.
- `setCategory(value: string)`
Устанавливает категорию товара и цвет фона в соответствии с категорией.
- `setImage(value: string)`
Устанавливает изображение товара.
- `setDescription(value: string | string[])`
Устанавливает описание товара.
- `setCheckPrice(item: LotItem)`
Проверяет наличие цены у товара и блокирует кнопку при ее отсутствии.

- `setSwitchButton(value: LotItem)`
Изменяет текст кнопки в зависимости от статуса товара (добавлен в корзину или нет).

Класс Page - Расширяет функциональность абстрактного базового класса Component

Управляет основным окном страницы

Методы:

- `setCounter(value: number)`
Устанавливает значение счетчика товаров в корзине.
- `setCatalog(items: HTMLElement[])`
Добавляет карточки товаров на страницу из переданного массива HTML-элементов.
- `setLocked(value: boolean)`
Блокирует или разблокирует прокрутку страницы.

Класс Basket - Расширяет функциональность абстрактного базового класса Component

класс представляет корзину на странице, отображает список выбранных товаров, их общую стоимость и предоставляет кнопку для оформления заказа.

Конструктор: Принимает контейнер (HTML-элемент) и объект событий (EventEmitter).
Инициализирует свойства класса, представляющие элементы корзины.

Методы:

- `setItems(items: HTMLElement[])`
Заменяет содержимое списка товаров в корзине на переданный массив HTML-элементов.
Если корзина пуста, выводит сообщение "Корзина пуста".
- `setSelected(items: string[] | object[])`
Устанавливает состояние кнопки оформления заказа в зависимости от выбранных товаров.
Если выбран хотя бы один товар, кнопка доступна для нажатия, в противном случае кнопка блокируется.
- `setTotal(total: number)`
Устанавливает общую стоимость товаров в корзине.

Класс Modal - Расширяет функциональность абстрактного базового класса Component

представляет модальное окно на странице.

Конструктор: Принимает контейнер (HTML-элемент), объект событий (IEvents) и объект страницы (IPage). Инициализирует свойства класса, представляющие кнопку закрытия и содержимое модального окна. Устанавливает слушатели событий для кнопки закрытия и фона модального окна.

Методы:

- `setContent(value: HTMLElement)`
Заменяет содержимое модального окна на переданный HTML-элемент.
- `open()`
Открывает модальное окно. Блокирует прокрутку основной страницы, добавляет класс для отображения модального окна и запускает событие "modal:open".
- `close()`
Закрывает модальное окно. Если передан объект страницы, разблокирует прокрутку основной страницы, удаляет класс для скрытия модального окна, удаляет содержимое модального окна и запускает событие "modal:close".
- `render(data: IModalData): HTMLElement`: Рендерит модальное окно с переданными данными, открывает его и возвращает его контейнер.

Класс `Form` - Расширяет функциональность абстрактного базового класса `Component`

используется для управления формами на веб-странице

Методы:

- `onInputChange(field: keyof T, value: string)`
Сообщает о том, что значение в инпуте было изменено, и запускает событие `formName.field:change`, передавая имя поля и его значение.
- `render(state?: Partial & IFormState)`
Рендерит состояние формы. Принимает частичное состояние формы и отображает его. Если состояние включает в себя данные об ошибках и валидности, отображает соответствующие изменения. Возвращает контейнер формы.

Класс `Order` является расширением класса `Form`

предназначен для управления формой оформления заказа. Он предоставляет методы для установки значений полей телефона и электронной почты.

Методы

- `phone(value: string)`
Устанавливает значение для поля телефона в форме заказа.
- `email(value: string)`
Устанавливает значение для поля электронной почты в форме заказа.

Класс `OrderPay` является расширением класса `Order`

предназначен для управления формой оформления заказа с возможностью выбора способа оплаты. Он добавляет функционал для обработки событий выбора способа оплаты.