10. Учебный проект: революция или эволюция? (Часть 1)

Рабочая ветка **module4-task1**

Задача

Классы

В этом задании мы начнем использовать объекты для описания компонентов приложения. А объекты будем строить с помощью классов.

1. Представим все наши компоненты в виде классов.

2. Функции, которые мы использовали для получения шаблона разметки, превратим в метод `getTemplate` класса. Этот метод по-прежнему должен возвращать разметку.

3. Перепишем передачу данных в разметку. Ранее мы брали данные из аргументов функции, теперь же мы будем брать данные из свойств класса, обращаясь через `this`. Но прежде их нужно туда записать. Передавать данные мы будем через параметры конструктора (при вызове с `new`), поэтому в описании метода `constructor` возьмем данные из аргументов и запишем их как **приватные** свойства класса.

4. Добавим метод `getElement`, который будет создавать DOM-элемент на основе шаблона, записывать его в приватное свойство класса `_element` и возвращать созданный DOM-элемент. Для этого потребуется описать вспомогательную функцию, например `createElement`. Лучше завести под такие функции отдельный файл, например `/src/utils.js`. А также нужно будет позаботиться о том, чтобы DOM-элемент создавался только в случае, когда он ещё не был создан.

5. И сразу же создадим метод `removeElement`. Он нам понадобится для очищения ресурсов. В нем мы должны удалить ссылку на созданный DOM-элемент. Для этого достаточно записать `null` в свойство класса `_element`.

6. Прежде, чем править наш код в `main.js`, нужно изменить функцию для рендеринга (вставки в DOM), которую мы написали в самом начале. Раньше это была функция-обертка над `insertAdjacentHTML`, которая принимала контейнер, шаблон и позицию отрисовки. Теперь вместо шаблона мы будем передавать DOM-элемент, поэтому `insertAdjacentHTML` нужно заменить на другую стандартную функцию, которая умеет вставлять DOM-элементы.

7. Теперь, когда подготовительные работы закончены, используйте в `main.js` для создания компонентов не

функции, а классы.

Настоящее редактирование

И в заключение мы реализуем замену карточки задачи на форму редактирования и обратно.

1. Избавьтесь от кода, который сразу отрисовывает первой в списке форму редактирования. Теперь при загрузке приложения должны отображаться только карточки задач.

2. В месте, где вы создаёте компонент карточки задачи, а потом его отрисовываете, создайте компонент формы редактирования. Теперь у вас есть два компонента: задача и форма редактирования задачи.

3. Найдите кнопку «Edit» в компоненте карточки задачи и тег `form` в компоненте формы редактирования (не нужно ходить за ними в `document`, используйте метод `getElement`). Навесьте на них пустые обработчики события `click` и события `submit` соответственно.

4. Реализуйте в добавленных обработчиках замену одного компонента на другой с помощью `replaceChild`.

12. Личный проект: революция или эволюция? (Часть 1)

Задача

Классы

В этом задании мы начнем использовать объекты для описания компонентов приложения. А объекты будем строить с помощью классов.

1. Представим все наши компоненты в виде классов.

2. Функции, которые мы использовали для получения шаблона разметки, превратим в метод `getTemplate` класса. Этот метод по-прежнему должен возвращать разметку.

3. Перепишем передачу данных в разметку. Ранее мы брали данные из аргументов функции, теперь же мы будем брать данные из свойств класса, обращаясь через `this`. Но прежде их нужно туда записать. Передавать данные мы будем через параметры конструктора (при вызове с `new`), поэтому в описании метода `constructor` возьмем данные из аргументов и запишем их как **приватные** свойства класса.

4. Добавим метод `getElement`, который будет создавать DOM-элемент на основе шаблона, записывать его в приватное свойство класса `_element` и возвращать созданный DOM-элемент. Для этого потребуется описать вспомогательную функцию, например `createElement`. Лучше завести под такие функции отдельный файл, например `/src/utils.js`. А также нужно будет позаботиться о том, чтобы DOM-элемент создавался только в случае, когда он ещё не был создан.

5. И сразу же создадим метод `removeElement`. Он нам понадобится для очищения ресурсов. В нем мы должны удалить ссылку на созданный DOM-элемент. Для этого достаточно записать `null` в свойство класса `_element`.

6. Прежде, чем править наш код в `main.js`, нужно изменить функцию для рендеринга (вставки в DOM), которую мы написали в самом начале. Раньше это была функция-обертка над `insertAdjacentHTML`, которая принимала контейнер, шаблон и позицию отрисовки. Теперь вместо шаблона мы будем передавать DOM-элемент, поэтому `insertAdjacentHTML` нужно заменить на другую стандартную функцию, которая умеет вставлять DOM-элементы.

7. Теперь, когда подготовительные работы закончены, используйте в `main.js` для создания компонентов не

функции, а классы.

Киноман

Настоящая информация о фильме

И в заключение мы реализуем показ и скрытие попапа с подробной информацией о фильме.

1. Избавьтесь от кода, который сразу отрисовывает попап с подробной информацией о фильме. Теперь при загрузке приложения должны отображаться только фильмы и блоки «Top rated» и «Most commented», если вы решили делать дополнительные пункты техзадания.

2. В месте, где вы создаёте компонент карточки фильма, а потом его отрисовываете, создайте компонент попапа. Теперь у вас есть два компонента: карточка фильма и попап.

3. Найдите обложку фильма, заголовок и элемент с количеством комментариев в компоненте карточки фильма и кнопку закрытия попапа (крестик) во втором компоненте (не нужно ходить за ними в `document`, используйте метод `getElement`). Навесьте на них пустые обработчики события `click`.

4. Реализуйте в добавленных обработчиках показ и скрытие попапа с подробной информацией о фильме с помощью `appendChild` и `removeChild`.

Bigtrip

Настоящее редактирование

И в заключение мы реализуем замену точки маршрута на форму редактирования и обратно.

1. Избавьтесь от кода, который сразу отрисовывает в списке форму редактирования. Теперь при загрузке приложения должны отображаться только точки маршрута.

2. В месте, где вы создаёте компонент точки маршрута, а потом его отрисовываете, создайте компонент формы редактирования. Теперь у вас есть два компонента: точка маршрута и форма редактирования точки маршрута.

3. Найдите кнопку раскрытия (стрелка вниз) в компоненте точки маршрута и тег `form` в компоненте формы редактирования (не нужно ходить за ними в `document`, используйте метод `getElement`). Навесьте на них пустые обработчики события `click` и события `submit` соответственно.

4. Реализуйте в добавленных обработчиках замену одного компонента на другой с помощью `replaceChild`.