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

Задание

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 их из аргументов и запишем 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 `createElement` (лучше завести под такие функции отдельный файл `/src/utils.js`). А так же 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. Напомню, что для этого достаточно записать `null` в свойство класса `_element`.

6. Прежде, чем править наш код в `main.js`, нужно подправить нашу функцию отрисовки, которую мы написали ранее (если вы её не написали, самое время ; -). 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 отрисовывать DOM-элементы.

7. Теперь, когда всё готово, в `main.js` для создания наших компонентов будем использовать не функции, а свежесозданные классы.

Замена карточки задачи на форму редактирования

И в заключении мы реализуем функциональность замены карточки задачи на форму редактирования и обратно.

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

2. В цикле, где вы создаете 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 первом компоненте и тег `form` во втором компоненте (кстати, не нужно ходить за ними в `document`, вы же описали метод `getElement` ; -) Навесьте на них пустые обработчики
события `click` и события `submit` соответственно.

4. Напишите реализацию для этих обработчиков, в которой реализуйте замену одного компонента на другой, используя `replaceChild`.

Выполненные пункты ТЗ

- Задачи в списке доступны только для просмотра. Для редактирования задачи пользователь нажимает на кнопку «Edit».

- Для редактирования задачи пользователь нажимает кнопку «Edit». В этот момент карточка задачи заменяется на форму редактирования задачи.

- В качестве значений по умолчанию используются фактические данные редактируемой задачи.

Задание можно отправить на проверку только после привязки к нему пулреквеста, отправленного из ветки `module4-task1`.

9. Учебный проект: революция или эволюция? (Часть 2)

Задание

В задании «Революция или эволюция? (Часть 1)» мы реализовали появление формы редактирования вместо карточки по нажатию кнопки «Edit» и обратно по отправке формы. И хотя мы пока не обрабатываем данные формы, в дальнейшем отправка будет означать сохранение, но сохранять данные требуется не всегда. Получается, что в настоящий момент у нас отсутствует возможность «закрыть» форму редактирования. Нужно это исправить.

1. В дополнение к обработчикам события `submit` на форме редактирования добавьте на страницу обработчик нажатия клавиши `Escape`, который будет, используя `replaceChild`, заменять один компонент на другой.

2. И в заключение мы добавим ещё одну вещь для большего удобства использования нашего приложения, а конкретно сообщение о том, что все задачи выполнены, если таковых нет или они все в архиве. Сообщение должно появляться вместо списка задач. Разметку сообщения вы найдете в папке `/markup`.

Выполненные пункты ТЗ

- Нажатие на кнопку `Esc` приводит к закрытию формы редактирования без сохранения изменений. Пользователь видит задачу в режиме просмотра.

- Если все задачи выполнены, то вместо задач отображается текст: `Click «ADD NEW TASK» in menu to create your first task`.

Задание можно отправить на проверку только после привязки к нему пулреквеста, отправленного из ветки `module4-task2`.

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

Задание

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 их 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 `createElement` (лучше завести под такие функции отдельный файл `/src/utils.js`). А так же 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. Напомню, что для этого достаточно записать `null` в свойство класса `_element`.

6. Прежде, чем править наш код в `main.js`, нужно подправить нашу функцию отрисовки, которую мы написали ранее (если вы её не написали, самое время ; -). 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 отрисовывать DOM-элементы.

7. Теперь, когда всё готово, в `main.js` для создания наших компонентов будем использовать не функции, а свежесозданные классы.

Большое путешествие

Замена точки маршрута на форму редактирования

И в заключении мы реализуем функциональность замены точки маршрута на форму редактирования и обратно.

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

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

3. Найдите кнопку со стрелкой вниз в первом компоненте и тег `form` во втором компоненте (кстати, не нужно ходить за ними в `document`, вы же описали метод `getElement` ; -) Навесьте на них пустые обработчики события `click` и события `submit` соответственно.

4. Напишите реализацию для этих обработчиков, в которой реализуйте замену одного компонента на другой, используя `replaceChild`.

Выполненные пункты ТЗ

- Для перехода к форме редактирования точки маршрута пользователь кликает по кнопке с изображением «Стрелка вниз» в правом углу карточки точки маршрута.

Задание можно отправить на проверку только после привязки к нему пулреквеста, отправленного из ветки `module4-task1`.

Киноман

Показ/скрытие попапа подробного описания фильма

И в заключении мы реализуем функциональность показа попапа с подробной информацией по фильму.

1. Избавьтесь от кода, который отрисовывает этот попап (например, в целях разработки) при старте приложения. Теперь при загрузке приложения должны отображаться только список с фильмами и блоки «Top rated» и «Most commented».

2. В цикле, где вы создаете 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 первом компоненте и кнопку закрытия во втором компоненте (кстати, не нужно ходить за ними в `document`, вы же описали метод `getElement`; -) Навесьте на них пустые обработчики события `click`.

4. Напишите реализацию для этих обработчиков, в которой реализуйте скрытие и показ попап с подробной информацией о фильме. Скрытие подразумевает удаление попапа из DOM.

Выполненные пункты ТЗ

- Клик по обложке фильма, заголовку, количеству комментариев открывает попап с подробной информацией о фильме.

Задание можно отправить на проверку только после привязки к нему пулреквеста, отправленного из ветки `module4-task1`.

11. Личный проект: революция или эволюция? (Часть 2)

Задание

Большое путешествие

В задании «Революция или эволюция? (Часть 1)» мы реализовали появление формы редактирования вместо точки маршрута по клику и обратно при отправке формы. И хотя мы пока не обрабатываем данные формы, в дальнейшем отправка будет означать сохранение, но сохранять данные требуется не всегда. Получается, что в настоящий момент у нас отсутствует возможность «закрыть» форму редактирования. Нужно это исправить.

1. В дополнение к обработчикам события `submit` на форме редатирования добавьте на страницу обработчик нажатия клавиши `Escape`, который будет, используя `replaceChild`, заменять один компонент на другой.
2. И в заключение мы добавим ещё одну вещь для большего удобства использования нашего приложения, а конкретно приглашение добавить первую точку маршрута, если таковые отсутствуют. Сообщение должно появляться вместо списка точек маршрута. Разметку сообщения вы найдете в папке `/markup`.

Выполненные пункты ТЗ

- При нажатии кнопки «Esc» форма создания/редактирования закрывается. Несохранённые изменения пропадают.
- В случае отсутствия точек маршрута вместо списка отображается текст: «Click New Event to create your first point».

  Задание можно отправить на проверку только после привязки к нему пулреквеста, отправленного из ветки `module4-task1`.

Киноман

В задании «Революция или эволюция? (Часть 1)» мы реализовали появление попапа с подробной информацией о фильме по клику на карточку фильма и его закрытие. Но в настоящий момент попап можно закрыть только кликнув на кнопку закрытия, что не всегда удобно. Нужно это исправить.

1. В дополнение к обработчикам события `click` в попапе добавьте на страницу обработчик нажатия клавиши `Escape`, который будет, используя `removeChild`, «закрывать» попап.
2. И в заключение мы добавим ещё одну вещь для большего удобства использования нашего приложения, а конкретно сообщение о том, что в системе пока нет фильмов. Сообщение должно появляться вместо списка фильмов. Разметку сообщения вы найдете в папке `/markup`.

Выполненные пункты ТЗ

- Попап можно закрыть нажатием на кнопку закрытия в правом верхнем углу (крестик) или нажатием на клавиатуре кнопки «Esc». При закрытии попап удаляется из DOM.
- В случае отсутствия фильмов вместо списка отображается текст: «There are no movies in our database».

  Задание можно отправить на проверку только после привязки к нему пулреквеста, отправленного из ветки `module4-task2`.