Домашние задание #4

#14 Учебный проект: одеть Надежду

В этом разделе мы опишем основные сценарии взаимодействия пользователя с сайтом

Задача

В файле`setup.js` опишите следующие сценарии взаимодействия пользователя с сайтом:

1. **Открытие/закрытие окна настройки персонажа:**
   - Окно`.setup` должно открываться по нажатию на блок`.setup-open`. Открытие окна производится удалением класса `hidden` у блока
   - Окно`.setup` должно закрываться по нажатию на элемент`.setup-close`, расположенный внутри окна

   Добавить обработчики для альтернативного ввода с клавиатуры `keydown` для кнопок открытия/закрытия диалога настройки персонажа:

   3. Когда иконка пользователя в фокусе `.setup-open-icon`, то окно настройки персонажа должно открываться по нажатию кнопки `ENTER`

      Не забудьте добавить `tabindex="0"` для иконки пользователя, чтобы она фокусировалась.

   4. Когда окно настройки персонажа открыто, нажатие на клавишу `ESC` должно закрывать диалог

      Если фокус находится на форме ввода имени, то окно закрываться не должно.

   5. Если окно открыто и фокус находится на кнопке закрытия окна, то нажатие клавиши `ENTER`должно приводить к закрытию диалога
   6. Если диалог открыт, нажатие на кнопку «Сохранить» приводит к отправке формы
   7. Если диалог открыт и фокус находится на кнопке «Сохранить», нажатие на `ENTER` приводит к отправке формы
2. **Валидация ввода имени персонажа.** Имя персонажа вводится в поле `.setup-user-name`. Добавьте следующие ограничения:
   - имя персонажа не может содержать менее 2 символов;
   - максимальная длина имени персонажа — 25 символов.

   Для указания ограничений на ввод используйте стандартные возможности форм HTML5.
3. **Изменение цвета мантии персонажа по нажатию.** Цвет мантии `.setup-wizard .wizard-coat` должен обновляться по нажатию на неё. Цвет мантии задаётся через изменение инлайнового CSS-свойства `fill` для элемента. Цвет должен сменяться произвольным образом на один из следующих цветов:
   - rgb(101, 137, 164)
   - rgb(241, 43, 107)
   - rgb(146, 100, 161)
   - rgb(56, 159, 117)
   - rgb(215, 210, 55)
   - rgb(0, 0, 0)

4. **Изменение цвета глаз персонажа по нажатию.** Цвет глаз волшебника меняется по нажатию на блок `.setup-wizard .wizard-eyes`. Возможные варианты цвета глаз персонажа:
    - black
    - red
    - blue
    - yellow
    - green
5. **Изменение цвета фаерболов по нажатию.** Цвет задаётся через изменение фона у блока `.setup-fireball-wrap`. Возможные варианты цвета:
    - #ee4830
    - #30a8ee
    - #5ce6c0
    - #e848d5
    - #e6e848

    Для того, чтобы на сервер отправились правильные данные, при изменении параметров персонажа должно изменяться и значение соответствующего скрытого инпута.

6. **Форма должна отправляться на урл `https://js.dump.academy/code-and-magick` методом `POST` с типом `multipart/form-data`**

Ограничений на алгоритм выбора цвета нет: это может быть случайный выбор, а может быть изменение цвета по порядку в списке.

Пример того, как может выглядеть персонаж после изменений:

#15 Личный проект: подробности

Перечитайте, пожалуйста, ТЗ

Начиная с этого задания мы будем плотно работать с техническим заданием проекта. Схема будет следующей: мы берём на выполнение задачи из ТЗ, номера пунктов будут указаны, и в задании разбираем некоторые детали реализации.

Задания являются дополнениями к ТЗ, объясняющими каким именно образом выполнять те или иные пункты, основным источником о том, как работает проект, является ТЗ.

Кексобукинг

В этом задании мы начнём реализацию сценария переключения режимов страницы между неактивным и активным, свяжем режим активации с отрисовкой новых меток на карте и сделаем так, чтобы при нажатии на каждую из меток на карте, показывалось объявление с подробной информацией.

Первое действие, которое нужно выполнить, перед тем, как приступить к этому заданию, вернуть страницу в исходное состояние. В прошлом разделе мы активировали карту, убрав у неё класс `.map--faded` и вызвали методы отрисовки похожих объявлений и метод отрисовки карточки. Проблема в том, что это не соответствует ТЗ — эти методы должны вызываться только в рамках соответствующих сценариев, поэтому мы уберём их вызовы, а самими методами воспользуемся позже. Пока что оставим в коде методы, созданные в прошлом задании, но саму страницу вернём в исходное состояние.

Еще нужно не забыть проверить пункт ТЗ, указывающий на то, что поля формы должны быть неактивны в исходном состоянии. В разметке проекта поля активны, поэтому их нужно отключить, т.е. добавить через DOM-операции или самим полям или `fieldset` которые их содержат, атрибут `disabled`.

1. Активация страницы

Страница Букинга может находиться в двух режимах: неактивном и активном.

В неактивном режиме страница находится сразу после открытия. В этом режиме отключены форма и карта и единственное действие, которое можно выполнить со страницей — перетащить метку адреса. Первое перетаскивание метки переводит страницу в активный режим.

Перетаскивание метки — это тема домашнего задания из следующего раздела, поэтому в этом разделе мы только сэмулируем перетаскивание. Любое перетаскивание состоит из трёх фаз: захвата элемента, его перемещения и отпускания элемента. На данном этапе нам достаточно описать реакцию на третью фазу: отпускание элемента. Для этого нужно добавить обработчик события `mouseup` на элемент `.map__pin--main`.

Обработчик события `mouseup` должен вызывать функцию, которая будет отменять изменения DOM-элементов, описанные в пункте «Неактивное состояние» технического задания.

Заполнение поля адреса

Кроме активации формы, перемещение метки приводит к заполнению поля адреса. В значении поля записаны координаты, на которые метка указывает своим острым концом. Поэтому в обработчике события `mouseup` на элементе метки, кроме вызова метода, переводящего страницу в активное состояние, должен находиться вызов метода, который устанавливает значения поля ввода адреса.

Ещё один момент заключается в том, что поле адреса должно быть заполнено всегда, в том числе сразу после открытия страницы. Насчёт определения координат метки в этом случае нет никаких инструкций, ведь в неактивном режиме страницы метка круглая, поэтому мы можем взять за исходное значение поля адреса середину метки. А при «перетаскивании» значение поля изменится на то, на которое будет указывать острый конец метки.
Для определения смещения координаты относительно левого верхнего угла метки можно использовать любой способ, в том числе, вычисление размеров метки. Кроме этого, можно хранить размеры метки как константу.

Просмотр подробной информации о похожих объявлениях

После перевода страницы в активный режим, нужно отрисовать на карте похожие объявления. Позже, в разделе про сеть, перевод в активный режим будет запускать загрузку объявлений, но пока что можно показать объявления сразу.
Нажатие на метку похожего объявления на карте, приводит к показу карточки с подробной информацией об этом объявлении. Получается, что для меток должны быть созданы обработчики событий, которые вызывают показ карточки с соответствующими данными.

Использовать ли делегирование

При решении этой задачи помните о том, что при клике на метку, нужно будет передавать в метод отрисовки карточки объект с данными, описывающими объявление. Если использовать для этой задачи делегирование, то нахождение этого объекта будет нетривиальным, потому что у вас будет использоваться один обработчик, у которого есть информация только о том, на каком DOM-элементе произошёл клик. Таким образом, если вы используете делегирование для решения этой задачи, вам нужно будет каким-то образом связать DOM-объекты с JS-объектами, которые их описывают.

Кекстаграм

В этом задании мы начнём реализацию сценария загрузки изображения и его редактирования, а также опишем показ фотографий в полноэкранном режиме.
Перед выполнением этого задания, нужно вернуть страницу в исходное состояние. Согласно ТЗ, оверлей `.big-picture`, показывающий фотографию в полноэкранном режиме показывается только по клику на уменьшенное изображение. В прошлом разделе вы выполняли задание, в котором показывали оверлей при загрузке страницы и заполняли его данными из первой сгенерированной фотографии. Теперь нам нужно приберечь этот код до поры: оставим в коде метод, который отрисовывает полноэкранный оверлей, но уберём его вызов, чтобы позже прописать его в одном (или не в одном) из обработчиков событий.

Загрузка изображения и показ формы редактирования

В этом проекте загрузка и редактирование настоящего изображения необязательны, поэтому в этом пункте можно ограничиться просто обработкой изменения значения поля выбора файла `#upload-file`. При наступлении события `change` на этом поле, можно сразу показывать форму редактирования изображения.
При написании обработчиков, реагирующих на закрытие формы, обратите внимание на то, что при закрытии формы, дополнительно нужно сбрасывать значение поля выбора файла `#upload-file`. В принципе, всё будет работать, если

при повторной попытке загрузить в поле другую фотографию,
но событие `change`не сработает, если вы попробуете загрузить ту же фотографию.

Применение эффекта для изображения и Редактирование размера изображения

Применение эффекта для изображения полностью программировать в этом задании не нужно. Уровень насыщенности эффекта изменяется перемещением ползунка, но перетаскивание — это тема следующей лекции и полностью реализовать перетаскивание у нас не получится, однако мы можем его сэмулировать.

Процесс перетаскивания состоит из трёх этапов: захват элемента, перемещение и отпускание. Два первых этапа мы разберём на будущей лекции, но пока что можем описать только отпускание. Для этого добавим
на пин слайдера `.scale__pin` обработчик события `mouseup`, который будет согласно ТЗ изменять уровень насыщенности фильтра для изображения. Для определения уровня насыщенности, нужно рассчитать положение пина слайдера относительно всего блока и воспользоваться пропорцией, чтобы понять, какой уровень эффекта нужно применить.

Обратите внимание, что при переключении фильтра, уровень эффекта должен сразу сбрасываться до начального состояния, т.е. логика по определению уровня насыщенности должна срабатывать не только при «перемещении» слайдера,
но и при переключении фильтров.

Показ изображения в полноэкранном режиме

После открытия страницы, нужно отрисовать пользовательские фотографии. Позже, в разделе про сеть, открытие страницы будет запускать загрузку фотографий, но пока что можно показать фотографии сразу.

Нажатие на фотографию приводит к показу фотографии в полноэкранном режиме. Получается, что для фотографий должны быть созданы обработчики событий, которые вызывают показ оверлея с соответствующими данными.

Использовать ли делегирование

При решении этой задачи помните о том, что при клике на фотографию, нужно будет передавать в метод отрисовки полноэкранной галереи объект с данными, описывающими объявление. Если использовать для этой задачи делегирование, то нахождение этого объекта будет нетривиальным, потому что у вас будет использоваться один обработчик, у которого есть информация только о том, на каком DOM-элементе произошел клик. Таким образом, если вы используете делегирование для решения этой задачи, вам нужно будет каким-то образом связать DOM-объекты с JS-объектами, которые их описывают.

#16 Личный проект: доверяй, но проверяй

Задача

В этом задании мы опишем сценарии взаимодействия пользователя с формой отправки данных и саму отправку. В случае Кексобукинга это будет форма размещения объявления, а в случае Кекстаграма — форма загрузки нового изображения.
Задание рассчитано на реализацию пунктов ТЗ, связанных с заполнением и отправкой формы на сервер. Для Кекстаграма это пункты 1, 2 и 3, для Кексобукинга — 1 и 2. В обоих проектах на данном этапе исключаются пункты, говорящие об отправке данных на сервер и соответствующей реакции: успешной и неуспешной отправке.
Больша́я часть работы в этом задании будет выполнена в разметке. Вам нужно проверить разметку вашего проекта и добавить соответствующие атрибуты и типы полей в разметку форм. Всем обязательным полям нужно добавить значение `required`. Проверить, правильные ли типы стоят у нужных полей и проставить этим полям дополнительные атрибуты, органичивающие длину значения, верхнюю и нижнюю границу и т. д.
В разметке задаётся и адрес, на который отправляются данные формы. В шестом разделе мы выполним задание, в котором мы перепишем механизм отправки данных, но пока что, достаточно убедиться, что у соответствующих тегов `form` прописаны правильные атрибуты.

Кекстаграм

Нетривиальный сценарий валидации в Кекстаграме — это хэш-теги. Для проверки валидности хэш-тегов, нужно вспомнить работу с массивами. Набор хэш-тегов можно превратить в массив, воспользовавшись методом `split`, который разбивает строки на массивы. После этого, вы можете написать цикл, который будет ходить по полученному массиву и проверять каждый из хэш-тегов на предмет соответствия ограничениям. Если хотя бы один из тегов не проходит нужных проверок, можно воспользоваться методом `setCustomValidity` для того, чтобы задать полю правильное сообщение об ошибке.

При решении этой задачи обратите внимание на то, что под длиной хэштега в 20 символов в ТЗ имеется ввиду длина, включающая символ решетки, поскольку решетка является частью тега.

Кексобукинг

Нетривиальный сценарий валидации: установка соответствия количества гостей количеству комнат. Для решения задачи, при желании, вы можете доработать разметку проекта. При решении этой задачи можно пойти несколькими путями. В любом случае, нужно будет подписаться на изменения значения поля количества комнат.

Первый подход заключается в том, чтобы физически ограничить возможность выбора неправильных вариантов. Для этого вы можете или удалять соответствующие элементы `option` из разметки или добавлять

им атрибут `disabled`. *Помните, что при таком подходе возникает проблема в сценарии, когда у пользователя уже выбран вариант, который вы хотите исключить, так как произойдет неявное изменение значения, которое пользователь не заметит*.

Второй подход заключается в использовании встроенного API для валидации и вызова метода `setCustomValidity` когда выбранное значение количества гостей не подходит под количество комнат.

Если форма заполнена верно, то должна показываться страница с успешно отправленными данными, если же форма пропустила какие-то некорректные значения, то будет показана страница с допущенными ошибками. В идеале у пользователя не должно быть сценария при котором он может отправить некорректную форму.