Чеклист для самопроверки.
9 Спринт.

Работа принимается

- HTML, CSS, JS-файлы и изображения должны быть в папке `src`.
- В проекте есть:
  - файл `index.html`;
  - директория `blocks`;
  - директория `pages` с файлами `index.css` и `index.js`;
  - папка `images` с картинками;
  - файлы скриптов `Card.js`, `FormValidator.js`, `UserInfo.js`, `Section.js`, `Popup.js`, `PopupWithForm.js`, `PopupWithImage.js`, `PopupWithConfirmation.js`, `Api.js` в директории `components`;
  - файл `README.md`;
  - файл `.gitignore` с исключениями `node_modules` и `dist`.
- Webpack настроен:
  - установлены `webpack`, `webpack-cli` и `webpack-dev-server`;
  - настроены сборки `build` и `dev`;
  - скрипты созданы в `package.json`;
  - настроена минификация и транспиляция JS-бабелем. Webpack собирает весь JavaScript в один файл и автоматически добавляет в HTML тег `script` со ссылкой на него;
  - настроена обработка CSS;
  - настроена минификация CSS и автоматическое добавление вендорных префиксов;
  - настроена обработка изображений и шрифтов;
  - если в HTML есть ссылки на локальные картинки, при сборке всё работает.
- Корректно задан `viewport`, прописаны `title` и `lang`.
- Стили подключены в отдельном файле.
- Код форматирован одинаково, соблюдается иерархия отступов.
- Файл `README.md` содержит:
  - заголовок-название;
  - описание проекта и его функциональности;
  - указание, что за технологии используются.
  - ссылку на GitHub Pages.
- Соблюдена методология БЭМ.
- Файловая структура по БЭМ.
- Вся функциональность работает корректно:
  - карточки, полученные с сервера, добавляются на страницу с помощью JS;
  - форма добавления карточки свёрстана, открывается, добавляет карточку;
  - работает нажатие на кнопку лайка;
  - удаление карточки реализовано корректно;
  - модальные окна закрываются на любых разрешениях экрана;
  - модальное окно с картинкой открывается, изображение не теряет пропорции;
  - реализовано плавное открытие и закрытие модального окна CSS-стилями;

    - для всех полей ввода в формах включена лайв-валидация;
    - кнопка отправки формы неактивна, если хотя бы одно из полей не проходит валидацию;

    - модальное окно закрывается по клику в любом месте вне этого окна и по нажатию на `Esc`.
- Проверка данных должна работать так:
    - данные любого поля ввода проверяются одной унифицированной функцией;
    - для проверки данных в поле используются HTML5-атрибуты и JS-свойство `ValidityState`;
    - за состояние кнопки сабмита отвечает отдельный метод класса;
    - за включение валидации формы отвечает метод класса `enableValidation`.
- Код объектно-ориентирован:
    - Использованы ES6-классы.
    - Каждый класс описан в отдельном JS-файле и импортирован в `index.js`.
    - Каждый класс выполняет строго одну задачу. Всё, что относится к решению этой задачи, находится в классе.
    - Для описания взаимодействия между классами используется слабое связывание, то есть внутри классов напрямую не создаются экземпляры других классов.
    - Экземпляр класса `Section` создаётся для каждого контейнера, в который требуется отрисовывать элементы. Класс соответствует описанию из проектной работы.
    - Экземпляр класса `Card` создаётся для каждой карточки. Класс соответствует описанию из проектной работы.
    - Экземпляр класса `FormValidator` создаётся для каждой проверяемой формы. Класс соответствует описанию из проектной работы.
    - Экземпляр класса `UserInfo` создается единожды. Класс соответствует описанию из проектной работы.
    - Экземпляр класса `Api` создается единожды. Класс соответствует описанию из проектной работы.
    - Класс `Popup` базовый, имеет трёх наследников, которые создаются для каждого модального окна. Класс и наследники соответствуют описанию из проектной работы. Чтобы подтвердить удаление карточки, добавлен ещё один вид попапа. Ему можно назначить обработчик сабмита, чтобы установить `id` карточки.
- Создан класс `Api`, внутри которого описаны запросы к серверу. Запросы к серверу не должны быть описаны внутри других классов или `index.js`
    - Каждый метод, включающий обращение к серверу содержит `return fetch`, т.е возвращает объект `Promise`
    - Все операции над DOM включены внутрь цепочки промисов.
    - Ответ от сервера всегда проверяется на корректность:

```
.then(res => {
  if (res.ok) {
    return res.json();
  }

  // если ошибка, отклоняем промис
  return Promise.reject(`Ошибка: ${res.status}`);
});
```

    - Каждый промис содержит обработку ошибок после обращения к серверу.
    - Внутри класса `Api` не создаются экземпляры других классов, не вызываются методы других классов. Используется слабое связывание между классами.
- При редактировании профиля текст кнопки меняется на: «Сохранение...», пока данные загружаются.
- При наведении указателя мыши на аватар, на нём должна появляться иконка редактирования. При клике открывается модальное окно.
- Карточки должны отображаться на странице только после получения `id` пользователя.

- При удалении карточки появляется модальное окно для подтверждения удаления. Модальное окно сделано из уже существующих компонент.
- У карточек отображается количество лайков.
- Реализованы все запросы к серверу, указанные в описании проектной работы.
- Модальное окно не закрывается, если кликнуть внутри него — по само́й форме, а не по оверлею.
- Слушатель событий, закрывающий модальное окно по нажатию на `Esc`, добавляется при открытии модального окна и удаляется при его закрытии.
- Если данные пришли от пользователя, нельзя передавать их свойству `innerHTML`.
- Код не повторяется. Если строчка нужна в нескольких местах, она должна быть вынесена в отдельную функцию.
- Если переменная объявлена через `let`, её значение должно быть напрямую перезаписано.
- Если переменная не перезаписывает своё значение, она объявлена через `const`.
- Нет «магических значений»: все числовые значения записаны в переменные.
- Операции над DOM-элементами выполняются до их вставки в разметку.
- Функции, декларированные как `function functionName() {}` (function declaration), должны быть вызваны после объявления.
- Поиск одного и того же DOM-элемента не должен происходить дважды.
- Функция выполняет только одну задачу, например, возвращает разметку карточки.
- Карточку можно добавить, нажав `Enter`, находясь в одном из текстовых полей;
- Кодстайл:
  - имена переменных и функций должны быть написаны в camelCase;
  - имена классов — существительные с прописной буквы;
  - имена переменных — существительные;
  - имена коллекций NodeList — существительные во множественном числе;
  - имя функции отражает то, что она делает.

  Для именования запрещены:
  - транслит;
  - неуместные сокращения.

Доступность интерфейса

- Все ссылки и интерактивные элементы имеют состояние наведения `:hover`.
- Контентные изображения имеют `alt` с корректным описанием, соответствующим языку страницы.

Работа отклоняется от проверки

- При открытии `index.html` в консоли возникают ошибки.
- Часть функциональности не реализована: отсутствует класс `Api`. Не описано более трёх методов класса `Api`.
- Работа содержит вопросы или просьбы о помощи к ревьюеру.
- На повторных итерациях не исправлены критические замечания.