# Чеклист для самопроверки. 7 Спринт.

## Работа принимается

- В проекте есть:
  - файлы `index.html`, `index.css`,
  - директория `blocks`,
  - папка `images` с картинками,
  - файлы скриптов `index.js`, `Card.js`, `FormValidator.js` и `README.md`.
- 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, указанная в задании, работает корректно и без ошибок:
  - 6 карточек создаются при помощи 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 отдельная функция;
  - за включение валидации формы отвечает функция `enableValidation`;
- Код объектно-ориентирован.
  - Использованы ES6-классы;
  - Каждый класс (`Card`, `FormValidator`) описан в отдельном JS-файле и импортирован в `index.js`;
  - Экземпляр класса `Card` создаётся для каждой карточки. Класс `Card` должен:
    - Принимать в конструктор ссылки на изображение и текст;
    - Принимать в конструктор селектор для `template`-элемента с шаблоном разметки;

        - Обладать приватными методами, которые установят слушателей событий, обработают клики, подготовят карточку к публикации;
        - Обладать публичным методом, который вернёт готовую разметку, с установленными слушателями событий.
    - Экземпляр класса `FormValidator` создаётся для каждой проверяемой формы. Этот класс должен:
        - Принимать в конструктор объект настроек с классами формы;
        - Принимать в конструктор ссылку на HTML-элемент проверяемой формы;
        - Содержать приватные методы для обработки формы. В методах за объектом настроек следует обращаться к полю класса, а не передавать его в каждый метод, как это было реализовано ранее;
        - Содержать публичный метод `enableValidation` — вызовите его после создания экземпляра класса.
    - Каждый класс выполняет строго одну задачу. Всё, что относится к решению этой задачи, находится в классе. Ни один другой класс к решению этой задачи не относится.
- 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 `Card` или `FormValidator`.
- Работа содержит вопросы или просьбы о помощи к ревьюеру.

- На повторных итерациях не исправлены критические замечания.