6. Учебный проект: время разбивать камни

Рабочая ветка **module2-task1**

В этом разделе мы разобьём `main.js` на отдельные JS-модули, чтобы избавиться от монолитного кода.

Задачи

Настроить сборку

1. Настройте сборку проекта так, чтобы зависимости в JS собирались при помощи сборщика модулей **Webpack**. Для этого выполните следующие шаги:

    1. Установите из npm пакеты `webpack` и `webpack-cli`.

    2. Создайте в корне проекта файл `webpack.config.js` и опишите конфигурацию сборки:

        - установите режим сборки для разработки;
        - задайте `main.js` точкой входа;
        - в качестве директории для сборки укажите папку `public`. Помните, что путь должен быть

абсолютный. Используйте `path.join`;

- файл сборки (бандл) назовите `bundle.js`;
- активируйте генерацию source-maps.

3. Добавьте в `package.json` скрипт с именем `build` для сборки кода с помощью **Webpack**.

2. В файле `public/index.html` вместо `./src/main.js` подключите `./bundle.js`.

3. Файлы сборки (именно файлы сборки, а не вся директория `public`) не должны попасть в репозиторий. Поэтому добавьте их в `.gitignore`.

4. Донастройте **ESLint** так, чтобы он работал с вашими файлами как с модулями ES2015, для этого укажите в начале файла `.eslintrc.yml`:

```
parserOptions:
    ecmaVersion: 2015
    sourceType: 'module'
```

Разбить на модули

1. Создайте модули (отдельные файлы) и перенесите в них написанные в прошлом задании функции для генерации DOM-элементов из `main.js`. Эти модули — компоненты. Поэтому:

    - для них нужно завести отдельную директорию `src/components`;

    - именуйте файлы как существительные (да, мы пока осознанно нарушаем критерий Б27);

    - для экспорта функций используйте именованный экспорт.

2. Импортируйте эти модули в `main.js`, сохранив логику работы.

3. Проверьте, что все хорошо и проект собирается выполнив `npm run build`.

В директории `public` должны появиться два файла: `bundle.js` и `bundle.js.map`. Так и есть? Тогда идём дальше. Если что-то пошло не так, то сравните вашу конфигурацию **Webpack** с конфигурацией на слайдах. Найдите несоответствия и устраните их.

Настроить сервер для разработки

Чтобы на каждое изменение кодовой базы не собирать бандл и не открывать `public/index.html` в браузере руками, настроим сервер для разработки.

1. Установите из npm пакет `webpack-dev-server`.

2. Опишите настройки сервера в `webpack.config.js`:

   - укажите абсолютный путь до директории со сборкой;

   - добавьте флаг слежки за изменением файлов, чтобы не обновлять страницу руками.

3. Осталось добавить npm-скрипт и можно пользоваться. Назовём его `start` со значением `webpack-dev-server --open`.

Теперь достаточно выполнить команду `npm start`. Браузер с нужным адресом откроется автоматически. Если этого не произойдёт, откройте адрес вручную. Адрес будет указан в терминале после запуска команды.