Тестовое задание по JavaScript

- Нельзя использовать jQuery или фреймворки, только ванильный JavaScript
- После выполнения задания залейте исходный код себе на GitHub, и задеплойте на GitHub Pages.
- Отправьте письмо с двумя ссылками (репо и живая страница) на `alexander.repeta@goit.ua`.

Задание 1

Напишите класс `StringBuilder(baseString)` и добавьте ему следующий функционал.

- Значение по умолчанию для параметра `baseSgtring` это пустая строка.
- У экземпляра будет свойство `value`, в которое записывается значение параметра `baseString`.
- Метод `append(str)` - получает парметр str (строку) и добавляет ее в конец свойства `value`.
- Метод `prepend(str)` - получает парметр str (строку) и добавляет ее в начало свойства `value`.
- Метод `pad(str)` - получает парметр str (строку) и добавляет ее в начало и в конец свойства `value`.

Следующий код должен выполнятся без ошибок.

```
const builder = new StringBuilder('.');

builder
  .append('^')
  .prepend('^')
  .pad('=');

console.log(builder); // '=^.^='
```

Задание 2

Напишите скрипт создания и очистки коллекции элементов. Пользователь вводит количество элементов в `input` и нажимает кнопку `Создать`, после чего рендерится коллекция. При нажатии на кнопку `Очистить`, коллекция элементов очищается.

Создайте функцию `createBoxes(amount)`, которая объявляет 1 параметр `amount` - число. Функция создает столько `div`, сколько указано в `amount` и добавляет их в `div#boxes`.

- Размеры самого первого div `30px` на `30px`
- Каждый следующий div после первого, должен быть шире и выше предыдущего на 10px
- Каждый созданный div должен иметь случайный rgb цвет фона

Создайте функцию `destroyBoxes()`, которая очищает `div#boxes`.

```
<div id="controls">
  <input type="number" min="0" max="100" step="1" class="js-input" />
  <button type="button" data-action="create">Создать</button>
```

```
  <button type="button" data-action="destroy">Очистить</button>
</div>

<div id="boxes"></div>
```

Задание 3

Напишите интерфейс и скрипт поиска и просмотра изображений по ключевому слову.

Интерфейс

Порма поиска создает DOM-дерево следующей структуры.

```
<form id="search-form">
  <input
    type="text"
    name="query"
    autocomplete="off"
    placeholder="Search images..."
  />
</form>
```

Галерея изображений создает DOM-дерево следующей структуры.

```
<ul>
  <!-- Набор элементов списка с карточками изображений -->
  <li>
    <!-- Карточка -->
    <a href="ссылка на большое изображение">
      <img
        src="ссылка на маленькое изображение"
        data-source="ссылка на большое изображение"
        alt="описание"
      />
    </a>
  </li>
</ul>
```

Pixabay API

Для HTTP-запросов используйте публичный Pixabay API. Pixabay поддерживает REST-пагинацию, пусть в ответе приходит по 20 элементов.

Каждое изобаржение описывается объектом, вам интересны следующие свойства:

- `webformatURL` - ссылка на маленькое изображение
- `largeImageURL` - ссылка на большое изображение

Бесконечный скрол

Добавьте функционал бесконечной прокрутки страницы. При скроле страницы, догружается и отрисовывается следующая порция изображений. Желательно сделать это вручную через Intersection Observer. В крайнем случае можно использовать библиотеку вроде Infinite Scroll.

Модальное окно

Добавьте возможность просмотра большой версии изображения в модальном окне при клике по изображению в галерее. Для модального окна используйте плагин basicLightbox.