## Лабораторная работа 2. Верстка на компоненты

**Полезные ссылки:**

1. [Подключение JavaScript кода](https://dvmn.org/encyclopedia/frontend/frontend_plug_js/)
2. [Типы данных JavaScript](https://developer.mozilla.org/ru/docs/Web/JavaScript/Data_structures)
3. [Выражения и операторы](https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Expressions_and_Operators)
4. [Условный (тернарный) оператор](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Conditional_operator)
5. [Циклы и итерации](https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Loops_and_iteration)
6. [Функции · Опыт программирования на javascript](https://kaineer.gitbooks.io/coding-in-javascript/content/functions.html#task)
7. [Функции - JavaScript | MDN](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Functions)
8. [**Отладка в браузере**](https://learn.javascript.ru/debugging-chrome)
9. [**Советы по стилю кода**](https://learn.javascript.ru/coding-style)
10. [Выразительный JavaScript: Формы и поля форм](https://habr.com/ru/articles/245731/)
11. [Техники валидации форм](https://htmlacademy.ru/blog/js/form-validation-techniques)
12. [Client-side form validation - Learn web development | MDN](https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation)
13. [Выразительный JavaScript: Обработка событий](https://habr.com/ru/articles/244041/)
14. [Метод EventTarget.addEventListener() - Интерфейсы веб API | MDN](https://developer.mozilla.org/ru/docs/Web/API/EventTarget/addEventListener)
15. [Делегирование событий в JavaScript | CAT.IN.WEB](https://www.cat-in-web.ru/delegirovanie-sobytij-v-javascript/)
16. [KeyboardEvent - Web APIs | MDN](https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent)
17. [События | Maps JavaScript API | Google for Developers](https://developers.google.com/maps/documentation/javascript/events?hl=ru)

## **Задание**

1. Прежде чем приступать к выполнению заданий необходимо сделать копию файла index.html. В папке проекта создайте папку markup и скопируйте туда файл index.html. В дальнейшем Вы будете использовать код разметки частями, и Вам придется удалить содержимое файла.
2. В папке проекта создадим папку src. В папке src создайте файл main.js.
3. Добавьте в папку src папку framework, а в ней файл render.js. Добавьте туда следующий код:

**const** RenderPosition = {

BEFOREBEGIN: 'beforebegin',

AFTERBEGIN: 'afterbegin',

BEFOREEND: 'beforeend',

AFTEREND: 'afterend',

};

**function** createElement(template) {

**const** newElement = document.createElement('div');

newElement.innerHTML = template;

return newElement.firstElementChild;

}

**function** render(component, container, place = RenderPosition.BEFOREEND) {

container.insertAdjacentElement(place, component.getElement());

}

export {RenderPosition, createElement, render};

1. Создайте модули (отдельные файлы) и опишите в них **классы** для генерации DOM-элементов. Эти модули — компоненты, или же View в рамках паттерна MVP. Поэтому:
   1. для них нужно завести отдельную директорию src/view;
   2. именуйте файлы как существительные;
   3. для экспорта классов используйте экспорт по умолчанию.

Компоненты, которые необходимо реализовать:

* 1. компонента заголовка (например, header-component.js)
  2. компонента для формы добавления задачи
  3. компонента для доски с задачами
  4. компонента для списка с задачами
  5. компонента для задачи

1. Теперь опишите в них классы для генерации DOM-элементов. Например, рассмотрим создание класса для компоненты заголовка и добавления новой задачи.

Файл header-component.js - компонента заголовка

import {createElement} from './render.js';

**function** createHeaderComponentTemplate() {

return (

`<header class="board-app\_\_header">

<div class="board-app\_\_inner">

<h1>Список задач</h1>

</div>

</header>`

);

}

export default **class** HeaderComponent {

getTemplate() {

return createHeaderComponentTemplate();

}

getElement() {

if (!this.element) {

this.element = createElement(this.getTemplate());

}

return this.element;

}

removeElement() {

this.element = null;

}

}

Разметку для функции createHeaderComponentTemplate возьмите из файла index.html.

1. Отрисуйте компоненты по одному экземпляру на страницу, кроме:
   1. компонент «Список задач» отрисовывается в списке 4 раза;
   2. компонент «Задача» отрисовывается 4 раза, но в каждом списке задач.

Для этого в файле main.js добавляете следующий код:

import HeaderComponent from './view/header-component.js';

import {render, RenderPosition} from './render.js';

**const** bodyContainer= document.querySelector('.board-app');

render(new HeaderComponent(), bodyContainer, RenderPosition.BEFOREBEGIN);

Имена css-классов у Вас будут отличаться.

В файле main.js класс .board-app - это класс, который относиться ко всем приложению и находится в теге body.

Основной файл index.html представляет собой следующий вид:

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="css/style.css">

<title>Список задач</title>

</head>

<body class="board-app">

<main class="board-app\_\_main">

<div class="board-app\_\_inner">

<section class="add-task">

</section>

<section class="taskboard">

</section>

</div>

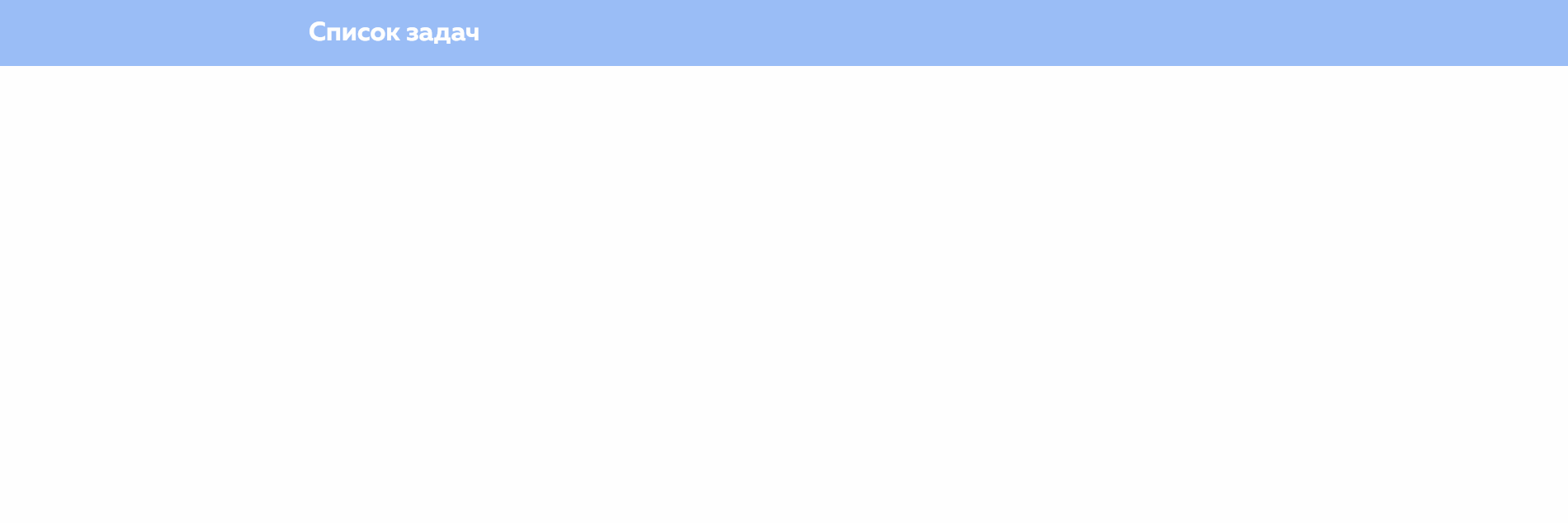
</main>

<script src="src/main.js" type="module"></script>

</body>

</html>

Запустить проект и убедитесь, что на страница выглядит следующим образом:



1. Далее добавим компонент формы добавления задачи аналогичным образом.

файл form-add-task-component.js

import {createElement} from './render.js';

**function** createFormAddTaskComponentTemplate() {

return (

`<form class="add-task\_\_form" aria-label="Форма добавления задачи">

<div class="add-task\_\_input-wrapper">

<label for="add-task">Новая задача</label>

<input type="text" name="task-name" id="add-task" placeholder="Название задачи..." required>

</div>

<button class="add-task\_\_button button" type="submit">

<svg width="22" height="22" viewBox="0 0 22 22" fill="none" xmlns="http://www.w3.org/2000/svg">

<rect x="10.0833" y="3.66663" width="1.83333" height="14.6667" fill="white" />

<rect x="18.3333" y="10.0833" width="1.83333" height="14.6667" transform="rotate(90 18.3333 10.0833)"

fill="white" />

</svg>

<span>Добавить</span>

</button>

</form>`

);

}

export default **class** FormAddTaskComponent {

getTemplate() {

return createFormAddTaskComponentTemplate();

}

getElement() {

if (!this.element) {

this.element = createElement(this.getTemplate());

}

return this.element;

}

removeElement() {

this.element = null;

}

}

Файл main.js

import HeaderComponent from './view/header-component.js';

import FormAddTaskComponent from './view/form-add-task-component.js';

import {render, RenderPosition} from './render.js';

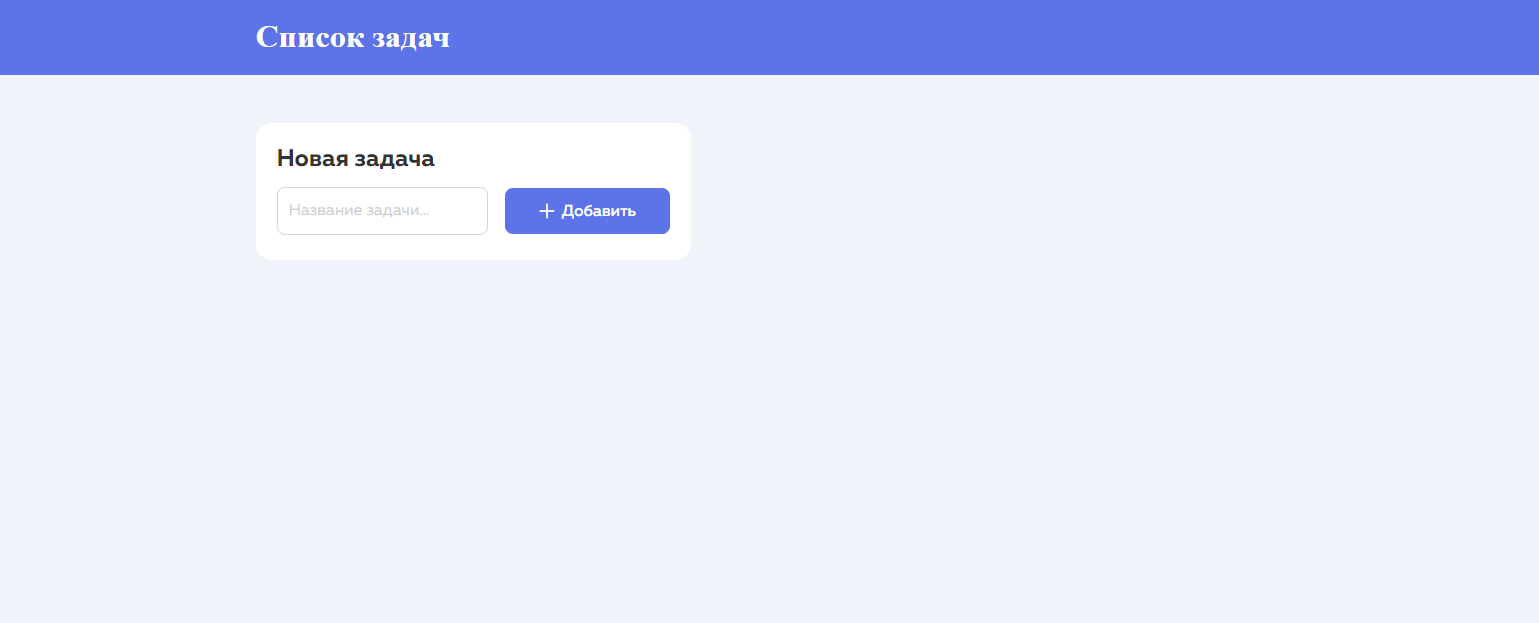
**const** bodyContainer = document.querySelector('.board-app');

**const** formContainer = document.querySelector('.add-task');

render(new HeaderComponent(), bodyContainer, RenderPosition.BEFOREBEGIN);

render(new FormAddTaskComponent(), formContainer);

Проверьте, что страница имеет вид:



1. Проделайте аналогичную работу с оставшимися компонентами. Должно получиться примерно следующее:

