МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №9**

по дисциплине: «Веб-технологии»

на тему: «Объединение JavaScript и CSS. FlexBox. Grid Layout. Движущиеся

элементы.»

Вариант 10

|  |  |
| --- | --- |
| Выполнил | Студент группы 22-ИТ-1  Короткевич П.А. |
|  |  |
| Проверил | Макарычева В.А |

Полоцк 2024

**Цель:** изучить принцип объединения JavaScript и CSS, разобраться в верстке страницы с помощью FlexBox и Grid, научиться применять полученные знания на практике.

**Задание 1**

Расположить блоки, используя Flexbox, на странице так, как показано на картинке.

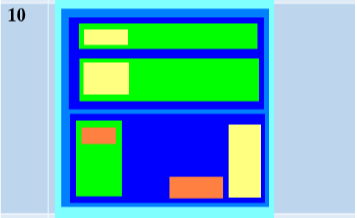


Рисунок 1 – Задание1

Листинг 1

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Flexbox Layout</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.container {

width: 600px;

height: 600px;

border: 10px solid rgb(52, 201, 247);

background-color: rgb(76, 76, 224);

display: flex;

justify-content: center;

align-items: center;

}

.inner-container {

display: flex;

flex-direction: column;

color :rgb(45, 167, 184);

justify-content: space-around;

align-items: center;

width: 80%;

height: 80%;

}

.blue-rectangle {

position: relative;

width: 540px;

height: 220px;

background-color: rgb(42, 11, 218);

}

.blue-rectangle::before,

.blue-rectangle::after {

content: "";

position: absolute;

width: 470px;

background-color: rgb(47, 214, 14);

}

.fourth-child{

position: absolute;

left: 52px;

top: 48px;

width: 140px;

height: 24px;

background-color: rgb(214, 214, 114);

}

.fifth-child{

position: absolute;

background-color: rgb(214, 214, 114);

width: 140px;

left: 52px;

height: 60px;

top: 110px;

z-index: 1;

}

.blue-rectangle::before {

top: 40px;

left: 40px;

height: 52px;

}

.blue-rectangle::after {

bottom: 40px;

left: 40px;

height: 80px;

}

.wide-blue-rectangle {

position: relative;

width: 540px;

height: 220px;

background-color: rgb(42, 11, 218);

}

.wide-blue-rectangle::before {

content: "";

width: 110px;

position: absolute;

height: 180px;

top: 20px;

left: 30px;

background-color: rgb(47, 214, 14);

}

.wide-blue-rectangle::after {

content: "";

position: absolute;

width: 80px;

height: 160px;

top: 40px;

right: 30px;

background-color: rgb(214, 214, 114);

}

.third-child{

position: absolute;

margin-top: 140px;

width: 140px;

height: 60px;

right: 130px;

background-color: rgb(233, 116, 7);

}

.sixth-child{

position: absolute;

background-color: rgb(233, 116, 7);

width: 90px;

height: 40px;

left: 40px;

top: 30px;

}

</style>

</head>

<body>

<div class="container">

<div class="inner-container">

<div class="blue-rectangle">

<div class="blue-rectangle::before">

<div class="fourth-child"></div>

</div>

<div class="blue-rectangle::after">

<div class="fifth-child"></div>

</div>

</div>

<div class="wide-blue-rectangle">

<div class="third-child"></div>

<div class="wide-blue-rectangle::before">

<div class="sixth-child"></div>

</div>

</div>

</div>

</div>

</body>

</html>

В данном HTML и CSS коде создается макет, используя flexbox для организации расположения элементов на веб-странице.

HTML разметка содержит основные элементы, такие как главный контейнер <div class="container">, который обеспечивает центрирование содержимого на странице и внутренний контейнер <div class="inner-container">, который служит для центрирования содержимого внутри главного контейнера.

Также в разметке присутствуют два прямоугольных блока: <div class="blue-rectangle"> и <div class="wide-blue-rectangle">, которые являются основными элементами макета.

В CSS стилях определены различные свойства для стилизации элементов и их позиционирования. Например, используются абсолютные позиции для размещения вложенных элементов внутри прямоугольных блоков.Также в CSS определены стили для главного контейнера, его размеров, а также для фона и границ элементов.Вложенные элементы имеют определенные размеры, цвет фона и позиции, чтобы создать заданный макет.

На следующем рисунке представлена реализация:

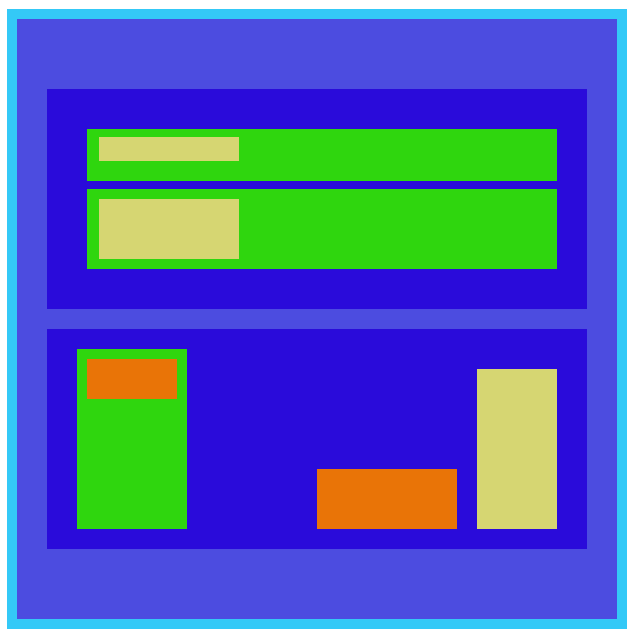


Рисунок 2 – Реализация первого задания

**Задание 2**

Реализовать CSS анимацию согласно варианту.

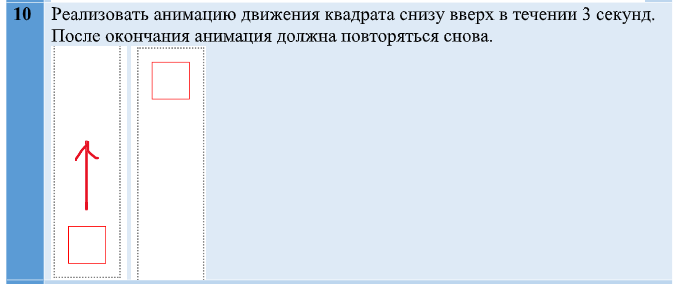


Рисунок 3 – Задание 2

Листинг 2

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Анимация движения квадрата</title>

<style>

#container {

position: relative;

height: 400px;

}

#square {

width: 50px;

height: 50px;

border: 2px solid red;

position: relative;

animation: moveUp 3s linear infinite;

}

@keyframes moveUp {

0% {

top: 100%;

}

100% {

top: 0;

}

}

</style>

</head>

<body>

<div id="container">

<div id="square"></div>

</div>

</body>

</html>

Этот HTML-код создает простую анимацию движения квадрата снизу вверх. Внутри контейнера <div id="container"> находится квадрат <div id="square"></div>. Квадрат имеет начальные размеры 50x50 пикселей и красную границу, он расположен внутри контейнера с помощью CSS свойства position: relative;, которое позволяет контролировать его положение относительно его обычного местоположения. Задана CSS анимация под названием moveUp, которая определяет движение квадрата снизу вверх. Анимация начинается с того момента, когда квадрат находится внизу контейнера (в 0% анимации) и заканчивается, когда он находится вверху контейнера (в 100% анимации). Длительность анимации составляет 3 секунды (3s), а скорость движения постоянная и линейная (linear). Свойство infinite указывает, что анимация должна повторяться бесконечно, то есть после завершения анимации квадрат снова начнет двигаться снизу вверх.

На следующем рисунке представлена реализация:

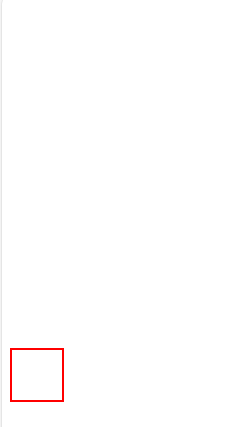


Рисунок 4 – Начальная точка

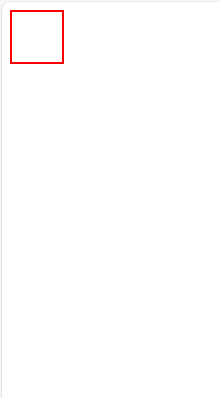


Рисунок 5 – Конечная точка

**Вывод:**

Изучение принципов объединения JavaScript и CSS, а также освоение FlexBox и Grid Layout, открывает широкие возможности для создания динамичных и гибких веб-страниц. При помощи JavaScript можно управлять стилями и анимациями элементов, реагировать на события пользователя и динамически изменять содержимое страницы. FlexBox и Grid Layout предоставляют мощные инструменты для гибкого и адаптивного размещения элементов на странице, что позволяет легко создавать сложные макеты с минимальным количеством кода. Понимание этих концепций позволяет создавать современные и интерактивные веб-приложения, а применение полученных знаний на практике помогает улучшить пользовательский опыт и повысить эффективность веб-разработки.