МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ ЕВФРОСИНИИ ПОЛОЦКОЙ»

Факультет информационных технологий

Кафедра технологий программирования

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине «Веб - технологии»

Тема: «Применение CSS-анимаций и переходов.»

|  |  |
| --- | --- |
| Выполнил: | студент группы 23-ИТ-1  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Ю. Страпко |
| Проверил: | ассистент кафедры ТП  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Макарычева |

|  |
| --- |
|  |

**Цель работы:**освоить CSS-анимации и трансформации для создания современных пользовательских интерфейсов.

1. **Краткая теория**

CSS-переходы позволяют анимировать исходное значение CSS-свойства на новое значение с течением времени, управляя скоростью смены значений свойств. Большинство свойств меняют свои значения за 16 миллисекунд, поэтому рекомендуемое время стандартного перехода – 200ms.

Смена свойств происходит при наступлении определенного события, которое описывается соответствующим псевдоклассом. Чаще всего используется псевдокласс: hover. Данный псевдокласс не работает на мобильных устройствах. Универсальным решением, работающим в настольных и мобильных браузерах, будет обработка событий с помощью JavaScript (например, переключение классов при клике).

Переходы применяются ко всем элементам, а также к псевдоэлементам :before и :after. Для задания всех свойств перехода обычно используют краткую запись свойства transition.

CSS-переходы могут применяться не ко всем свойствам и их значениям. Свойство transition-property содержит название CSS-свойств, к которым будет применен эффект перехода. Значение свойства может содержать как одно свойство, так и список свойств через запятую. При создании перехода можно использовать как начальное, так и конечное состояние элемента.

Создаваемые эффекты должны быть ненавязчивыми. Не все свойства требуют плавного изменения во времени, что связано с пользовательским опытом. Например, при наведении на ссылку мы хотим видеть мгновенную смену цвета ссылки или цвета и стиля подчёркивания. Поэтому переходы следует использовать для тех свойств, к которым действительно нужно привлечь внимание

1. **Практическая часть**

**Задание:** Реализуйте следующие эффекты:

− Плавные переходы (transition) при наведении на ссылки и кнопки.

− Анимацию появления блока текста (keyframes), например, с эффектом увеличения и прозрачности.

− Трансформацию изображений (вращение, увеличение при наведении).

Дополнительно: используйте @keyframes для создания сложной анимации, например, анимации логотипа или иконки загрузки.

Код главной HTML – страницы представлен в листинге 2.1.

Листинг 2.1 – файл styles.css.

1. header {
2. background-color: #273d53;
3. padding: 15px;
4. display: flex;
5. flex-direction: row;
6. justify-content: space-around;
7. align-items: center;
8. }
9. header a {
10. color: #ecf0f1;
11. text-decoration: none;
12. padding: 10px 15px;
13. border-radius: 5px;
14. transition: background-color 0.5s ease;
15. }
16. header a:hover {
17. background-color: #4d6780;
18. }
19. main{
20. display: flex;
21. flex-direction: column;
22. margin: 10px;
23. gap: 10px;
24. }
25. .diva {
26. background-color: #273d53;
27. padding: 15px;
28. display: flex;
29. flex-direction: row;
30. justify-content: space-around;
31. align-items: center;
32. }
33. .diva a{
34. color: #ecf0f1;
35. text-decoration: none;
36. padding: 10px 15px;
37. border-radius: 5px;
38. transition: background-color 5.5s ease;
39. }
40. .diva a:hover {
41. background-color: #4d6780;
42. }
43. button{
44. border: none;
45. background-color: rgb(0, 0, 0);
46. color: #ffffff;
47. }
48. .bone{
49. transition: 1.5s ease;
50. }
51. .bone:hover{
52. background-color: rgb(104, 19, 19);
53. }
54. .btwo{
55. background-color: blue;
56. color: white;
57. padding: 20px;
58. border: none;
59. border-radius: 5px;
60. transition: 0.3s ease;
61. }
62. .btwo:hover {
63. background-color: rgb(0, 139, 32);
64. transform: scale(0.5);
65. }
66. .bthree{
67. transition: 0.8s ease;
68. }
69. .bthree:hover{
70. opacity: 0.3;
71. }
72. @keyframes move{
73. 0% {opacity: 0; }
74. 25% {opacity: 0.25; }
75. 50% {opacity: 0.5; }
76. 75% {opacity: 0.75; }
77. 100% {opacity: 1; }
78. }
79. .pone{
80. animation: move 3s;
81. }
82. @keyframes scale{
83. 0% {scale: 0.5;}
84. 25% {scale: 0.75;}
85. 50% {scale: 1;}
86. 75% {scale: 1.25;}
87. 100% {scale: 1.5;}
88. }
89. .ptwo{
90. animation: scale 3s;
91. }
92. img{
93. width: 150px;
94. height: 100px;
95. transition: 0.8s ease;
96. margin: 10px;
97. }
98. .ione:hover{
99. transform: scale(1.5);
100. }
101. .itwo:hover{
102. transform: scale(0.5);
103. }
104. .ithree:hover{
105. transform: rotate(360deg);
106. }

В данном листинге были использованы следующие html-тэги:

* flex-direction: row; – элементы выстраиваются в строку (горизонтально);
* justify-content: space-around; – равномерно распределяет элементы по горизонтали с отступами по краям;
* transition: background-color 0.5s ease; – добавляет плавный переход для изменения фона за 0.5 секунды;
* transform: scale(0.5); – (при наведении) — уменьшает кнопку в 2 раза;
* opacity: 0.3; – (при наведении) — уменьшает прозрачность до 30%;
* @keyframes move – определяет анимацию с именем move. Анимация описывает, как изменяются свойства элемента (например, прозрачность) с течением времени;
* @keyframes scale – определяет анимацию с именем scale. Эта анимация изменяет масштаб (размер) элемента с течением времени. В данном случае анимация плавно увеличивает масштаб элемента от 0.5 (половина размера) до 1.5 (полтора размера);
* animation: scale 3s; – применяет анимацию с именем scale к элементу, и эта анимация длится 3 секунды;
* transform: scale(1.5); – применяет масштабирование к элементу, увеличивая его размер в 1.5 раза относительно исходного размера;

**3.Контрольные вопросы**

**1. Для чего нужны CSS-переходы?**

CSS-переходы позволяют плавно изменять значения свойств элемента (например, цвет, размер, положение) при изменении состояния (например, при наведении курсора или активации). Они делают интерфейс более динамичным и визуально привлекательным.

**2. К каким элементам можно применить CSS-переходы?**

CSS-переходы можно применить к любому элементу, который поддерживает CSS-свойства, изменяемые анимированно (например, color, background-color, width, height, transform и т.д.).

**3. Опишите свойство transition-duration.**

Свойство transition-duration определяет длительность перехода. Оно задает, сколько времени займет анимация изменения свойства. Например:

transition-duration: 0.5s; /\* Переход длится 0.5 секунды \*/

**4. Опишите свойство transition-timing-function.**

Свойство transition-timing-function определяет функцию плавности перехода. Оно задает, как будет изменяться скорость анимации. Например:

transition-timing-function: ease; /\* Плавное ускорение и замедление \*/

Другие значения: linear, ease-in, ease-out, ease-in-out, cubic-bezier().

**5. Опишите свойство transition-delay.**

Свойство transition-delay задает задержку перед началом перехода. Например:

transition-delay: 1s; /\* Переход начнется через 1 секунду \*/

**6. Для чего нужны CSS-анимации?**

CSS-анимации позволяют создавать сложные и многоэтапные анимации, которые могут повторяться, изменять направление и управляться с помощью ключевых кадров (@keyframes). Они используются для создания динамичных эффектов, таких как вращение, перемещение, изменение размера и т.д.

**7. К каким элементам можно применить CSS-анимацию?**

CSS-анимации можно применить к любому элементу, который поддерживает CSS-свойства, изменяемые анимированно (например, transform, opacity, color и т.д.).

**8. Опишите правило @keyframes.**

Правило @keyframes определяет ключевые кадры анимации. Оно задает, как изменяются свойства элемента на разных этапах анимации. Например:

@keyframes move {

0% { transform: translateX(0); }

100% { transform: translateX(100px); }

}

**9. Опишите свойство animation-duration.**

Свойство animation-duration определяет длительность анимации. Например:

animation-duration: 2s; /\* Анимация длится 2 секунды \*

**10. Опишите свойство animation-timing-function.**

Свойство animation-timing-function определяет функцию плавности анимации. Оно задает, как будет изменяться скорость анимации. Например:

animation-timing-function: ease-in-out; /\* Плавное ускорение и замедление \*/

**11. Опишите свойство animation-iteration-count.**

Свойство animation-iteration-count определяет количество повторений анимации. Например:

animation-iteration-count: 3; /\* Анимация повторится 3 раза \*/

Значение infinite заставляет анимацию повторяться бесконечно.

**12. Опишите свойство animation-direction.**

Свойство animation-direction определяет направление анимации. Например:

animation-direction: alternate; /\* Анимация чередуется вперед и назад \*/

Другие значения: normal, reverse, alternate-reverse.

**13. Опишите свойство animation-play-state.**

Свойство animation-play-state определяет состояние анимации (воспроизведение или пауза). Например:

animation-play-state: paused; /\* Анимация на паузе \*/

**14. Опишите свойство animation-delay.**

Свойство animation-delay задает задержку перед началом анимации. Например:

animation-delay: 1s; /\* Анимация начнется через 1 секунду \*/

**15. Опишите свойство animation-fill-mode.**

Свойство animation-fill-mode определяет, как элемент выглядит до и после анимации. Например:

animation-fill-mode: forwards; /\* Элемент сохранит стили последнего кадра анимации \*/

Другие значения: backwards, both, none.

**16. Для чего нужны CSS-трансформации?**

CSS-трансформации позволяют изменять внешний вид элемента, применяя к нему такие эффекты, как масштабирование, вращение, перемещение и наклон. Они используются для создания визуальных эффектов без изменения макета.

**17. К каким элементам применяются CSS-трансформации?**

CSS-трансформации применяются к любым элементам, которые поддерживают CSS-свойства (например, div, img, button и т.д.).

**18. Какие виды CSS-трансформации существуют?**

Основные виды трансформаций:

1. translate() — перемещение элемента.
2. rotate() — вращение элемента.
3. scale() — масштабирование элемента.
4. skew() — наклон элемента.
5. matrix() — комбинированная трансформация.

**19. Опишите свойство transform и его значения.**

Свойство transform применяет трансформации к элементу. Например:

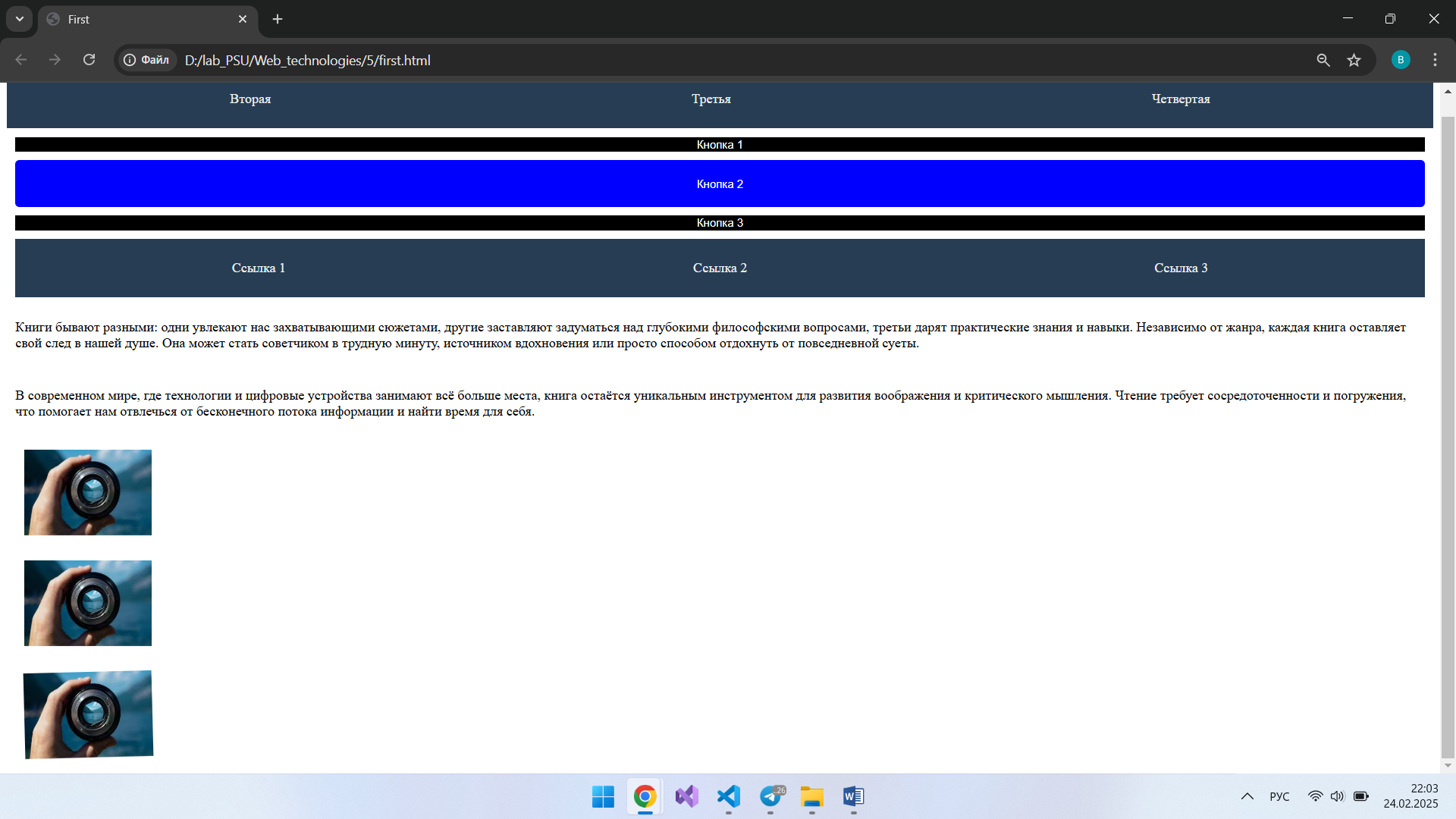
transform: rotate(45deg) scale(1.5); /\* Вращение на 45 градусов и увеличение в 1.5 раза \*/

**20. Опишите свойство transform-origin.**

Свойство transform-origin определяет точку преобразования (откуда начинается трансформация). Например:

transform-origin: top left; /\* Трансформация начинается от верхнего левого угла \*/

**4.Скриншоты результата**



**Рисунок 4.1 – Главная страница**

**Вывод:** в ходе данной лабораторной работы были изучены CSS-анимации и трансформации для создания современных пользовательских интерфейсов.