ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИЙ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ

ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Курсовая работа

Нукус 2015

Содержание:

I. Введение
II. Основная часть

1. Правило @keyframes
2. Название анимации animation-name
3. Продолжительность анимации animation-duration
4. Временная функция animation-timing-function
5. Анимация с задержкой animation-delay
6. Повтор анимации animation-iteration-count
7. Направление анимации animation-direction
8. Краткая запись анимации animation
9. Проигрывание анимации animation-play-state
10. Состояние элемента до и после воспроизведения анимации animation-fill-mode
11. Множественные анимации
12. CSS переходы
13. CSS трансформации
14. Примеры CSS анимации

III. Заключение
IV. Список использованной литературы

Введение

Фронтенд-дизайнеры и разработчики используют CSS для создания стилей, позиционирования элементов — в общем, для того, чтобы сайты выглядели красиво. Часто CSS применяют и для добавления движений на веб-страницы, хотя дело обычно не идёт дальше плавных переходов или анимации.

На самом деле анимация может быть мощным инструментом, который позволяет посетителям сайта и клиентам легче взаимодействовать с интерфейсом продукта и быстрее добиваться своих целей. Этого можно достичь, если следовать в своей работе некоторым основополагающим принципам.

CSS3 анимация придаёт сайтам динамичность. Она оживляет веб-страницы, улучшая взаимодействие с пользователем. В отличие от CSS3 переходов, создание анимации базируется на ключевых кадрах, которые позволяют автоматически воспроизводить и повторять эффекты на протяжении заданного времени, а также останавливать анимацию внутри цикла.

CSS3 анимация может применяться практически для всех html-элементов, а также для псевдоэлементов :before и :after. При создании анимации не стоит забывать о возможных проблемах с производительностью, так как на изменение некоторых свойств требуется много ресурсов.

Также не следует использовать анимацию только потому, что вы подобные эффекты нравятся вам. Неуместная анимация будет раздражать пользователей и мешать им. Анимация должна поддерживать взаимодействие с пользователем. Например, можно добавить небольшой эффект (отблеск или отскок) при нажатии на кнопку, чтобы подтвердить взаимодействие.

В этой курсовой работе рассматривается основные возможности CSS для создания и просмотра анимации в веб страницах. Рассматриваются такие свойства как

1. Правило @keyframes

Создание анимации начинается с установки **ключевых кадров** правила @keyframes. Кадры определяют, какие свойства на каком шаге будут анимированы. Каждый кадр может включать один или более блоков объявления из одного или более пар свойств и значений. Правило @keyframes содержит имя анимации элемента, которое связывает правило и блок объявления элемента.

```
1     @keyframes shadow {
2     from {
3       text-shadow: 0 0 3px black;
4     }
5     50% {
6       text-shadow: 0 0 30px black;
7     }
8     to {
9       text-shadow: 0 0 3px black;
10    }
11    }
```

Ключевые кадры создаются с помощью ключевых слов from и to (эквивалентны значениям 0% и 100%) или с помощью процентных пунктов, которых можно задавать сколько угодно. Также можно комбинировать ключевые слова и процентные пункты. Если кадры имеют одинаковые свойства и значения, их можно объединить в одно объявление:

```
1     @keyframes move {
2     from, to {
3     top: 0;
4     left: 0;
5     }
6     25%, 75% {
7     top: 100%;
8     }
9     50% {
10    top: 50%;
11    }
      }
```

Если 0% или 100% кадры не указаны, то браузер пользователя создает их, используя вычисляемые (первоначально заданные) значения анимируемого свойства. Если у двух ключевых кадров будут одинаковые селекторы, то последующий отменит действие предыдущего.

После объявления правила @keyframes, мы можем ссылаться на него в свойстве animation:

```
1    h1 {
2    font-size: 3.5em;
3    color: darkmagenta;
```

```
4     animation: shadow  2s infinite ease-in-out;
5     }
```

Не рекомендуется анимировать нечисловые значения (за редким исключением), так как результат в браузере может быть непредсказуемым. Также не следует создавать ключевые кадры для значений свойств, не имеющих средней точки, например, для значений свойства color: pink и color: #ffffff, width: auto и width: 100px или border-radius: 0 и border-radius: 50% (в этом случае правильно будет указать border-radius: 0%).

2.Название анимации animation-name

Свойство задаёт имя анимации. Имя анимации создаётся в правиле @keyframes. Рекомендуется использовать название, отражающее суть анимации, при этом можно использовать одно или несколько слов, связанных между собой при помощи пробела - или символа нижнего подчеркивания _. Свойство не наследуется.

animation-name	
Значения:	
имя анимации	Имя анимации, которое связывает правило @keyframes с селектором.
none	Значение по умолчанию, означает отсутствие анимации. Также используется, чтобы отменить анимацию элемента из группы элементов, для которых задана анимация.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
1     div {
2     animation-name: mymove;
3     }
```

3. Продолжительность анимации animation-duration

Свойство устанавливает продолжительность анимации, задаётся в секундах или миллисекундах, отрицательные значения не допустимы. Не наследуется.

Если для элемента задано более одной анимации, то можно установить разное время для каждой, перечислив значения через запятую.

animation-duration	
Значения:	
время	Длительность анимации задается в секундах s или миллисекундах ms.
initial	Устанавливает значение свойства в значение по умолчанию 0.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
animation-duration: 2s;
}
```

4. Временная функция animation-timing-function

Свойство определяет изменение скорости от начала до конца анимации с помощью временных функций. Задаётся при помощи ключевых слов или кривой Безье cubic-bezier(x1, y1, x2, y2). Не наследуется. График проигрования анимации приведён на рис.1

animation-timing-function	
Значения:	
ease	Функция по умолчанию, анимация начинается медленно, разгоняется быстро и замедляется в конце. Соответствует cubic-bezier(0.25,0.1,0.25,1).
linear	Анимация происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует cubic-bezier(0,0,1,1).
ease-in	Анимация начинается медленно, а затем плавно ускоряется в конце. Соответствует cubic-bezier(0.42,0,1,1).

ease-out	Анимация начинается быстро и плавно замедляется в конце. Соответствует cubic-bezier(0,0,0.58,1).
ease-in-out	Анимация медленно начинается и медленно заканчивается. Соответствует cubic-bezier(0.42,0,0.58,1).
cubic-bezier(x1, y1, x2, y2)	Позволяет вручную установить значения от 0 до 1. На этом сайте вы сможете построить любую траекторию скорости изменения анимации.
step-start	Задаёт пошаговую анимацию, разбивая анимацию на отрезки, изменения происходят в начале каждого шага. Эквивалентно steps(1, start).
step-end	Пошаговая анимация, изменения происходят в конце каждого шага. Эквивалентно steps(1, end).
steps(количество шагов,start|end)	Ступенчатая временная функция, которая принимает два параметра. Количество шагов задается целым положительным числом. Второй параметр необязательный, указывает момент, в котором начинается анимация. Со значением start анимация начинается в начале каждого шага, со значением end — в конце каждого шага с задержкой. Задержка вычисляется как результат деления времени анимации на количество шагов. Если второй параметр не указан, используется значение по умолчанию end.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

рис.1 Вид времени выполнения основных значении animation-timing-function

Синтаксис

```
div {
animation-timing-function: linear;
}
```

С помощью пошаговой анимации можно создавать интересные эффекты, например, печатающийся текст или индикатор загрузки.

5. Анимация с задержкой animation-delay

Свойство игнорирует анимацию заданное количество времени, что даёт возможность по отдельности запускать каждую анимацию. Отрицательная задержка начинает анимацию с определенного момента внутри её цикла, т.е. со времени, указанного в задержке. Это позволяет применять анимацию к нескольким элементам со сдвигом фазы, изменяя лишь время задержки.

Чтобы анимация началась с середины, нужно задать отрицательную задержку, равную половине времени, установленном в animation-duration. Не наследуется.

animation-delay	
Значения:	
время	Задержка анимации задается в секундах s или миллисекундах ms. Значение по умолчанию 0.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
animation-delay: 2s;
}
```

6. Повтор анимации animation-iteration-count

Свойство позволяет запустить анимацию несколько раз. Значение 0 или любое отрицательное число удаляют анимацию из проигрывания. Не наследуется.

animation-iteration-count	
Значения:	
число	С помощью целого числа задается количество повторов анимации. Значение по умолчанию 1. Дробные значения больше 1 будут воспроизводить анимацию, обрезая её на части следующей итерации.
infinite	Анимация проигрывается бесконечно.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
animation-iteration-count: 3;
}
```

7. Направление анимации animation-direction

Свойство задает направление повтора анимации. Если анимация повторяется только один раз, то это свойство не имеет смысла. Не наследуется.

animation-direction	
Значения:	
alternate	Анимация проигрывается с начала до конца, затем в обратном направлении.
alternate-reverse	Анимация проигрывается с конца до начала, затем в обратном направлении.
normal	Значение по умолчанию, анимация проигрывается в обычном направлении, с начала и до конца.

reverse	Анимация проигрывается с конца.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
    animation-direction: alternate;
}
```

8. Краткая запись анимации

Все параметры воспроизведения анимации можно объединить в одном свойстве — animation, перечислив их через пробел:

```
animation: animation-name animation-duration
animation-timing-function animation-delay animation-
iteration-count animation-direction;
```

Для воспроизведения анимации достаточно указать только два свойства — animation-name и animation-duration, остальные свойства примут значения по умолчанию. Порядок перечисления свойств не имеет значения, единственное, время выполнения анимации animation-duration обязательно должно стоять перед задержкой animation-delay.

9. Проигрывание анимации animation-play-state

Свойство управляет проигрыванием и остановкой анимации. Остановка анимации внутри цикла возможна через использование этого свойства в скрипте JavaScript. Также можно останавливать анимацию при наведении курсора мыши на объект — состояние :hover. Не наследуется.

animation-play-state	
Значения:	
paused	Останавливает анимацию.

running	Значение по умолчанию, означает проигрывание анимации.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div:hover {
animation-play-state: paused;
}
```

10. Состояние элемента до и после воспроизведения анимации animation-fill-mode

Свойство определяет порядок применения определенных в @keyframes стилей к объекту. Не наследуется.

animation-fill-mode	
Значения:	
none	Значение по умолчанию. Состояние элемента не меняется до или после воспроизведения анимации.
forwards	Воспроизводит анимацию до последнего кадра по окончанию последнего повтора и не отматывает ее к первоначальному состоянию.
backwards	Возвращает состояние элемента после загрузки страницы к первому кадру, даже если установлена задержка animation-delay, и оставляет его там, пока не начнется анимация.
both	Позволяет оставлять элемент в первом ключевом кадре до начала анимации (игнорируя положительное значение задержки) и задерживать на последнем кадре до конца последней анимации.

initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
animation-fill-mode: forwards;
}
```

11. Множественные анимации

Для одного элемента можно задавать несколько анимаций, перечислив их названия через запятую:

```
div {animation: shadow 1s ease-in-out 0.5s alternate, move 5s linear 2s;}
```

CSS3 переходы

CSS3 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, таких как iPhone или Android.

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

Название свойства 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. Свойство не наследуется.

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.

transition-property	
Значения:	
none	Отсутствие свойства для перехода.
all	Значение по умолчанию. Применяет эффект перехода ко всем свойствам элемента.
свойство	Определяет список css-свойств, перечисленных через запятую, участвующих в переходе.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
width: 100px;
transition-property: width;
}
div:hover {
width: 300px;
}
```

Продолжительность перехода transition-duration

Задаёт промежуток времени, в течение которого должен осуществляться переход. Если разные свойства имеют разные значения для перехода, они указываются через запятую. Если продолжительность перехода не указана, то анимация при смене значений свойств происходить не будет. Свойство не наследуется.

transition-duration	
Значения:	
время	Время перехода указывается в секундах или миллисекундах, например, 2s или 5ms.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
transition-duration: .2s;
```

Функция перехода transition-timing-function

Свойство задаёт временную функцию, которая описывает скорость перехода объекта от одного значения к другому. Если вы определяете более одного перехода для элемент, например, цвет фона элемента и его положение, вы можете использовать разные функции для каждого свойства. Свойство не наследуется.

transition-timing-function
Значения:

ease	Функция по умолчанию, переход начинается медленно, разгоняется быстро и замедляется в конце. Соответствует cubic-bezier(0.25,0.1,0.25,1).
linear	Переход происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует cubic-bezier(0,0,1,1).
ease-in	Переход начинается медленно, а затем плавно ускоряется в конце. Соответствует cubic-bezier(0.42,0,1,1).
ease-out	Переход начинается быстро и плавно замедляется в конце. Соответствует cubic-bezier(0,0,0.58,1).
ease-in-out	Переход медленно начинается и медленно заканчивается. Соответствует cubic-bezier(0.42,0,0.58,1).
cubic-bezier(x1, y1, x2, y2)	Позволяет вручную установить значения от 0 до 1 для кривой ускорения. На этом сайте вы сможете построить любую траекторию перехода.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
transition-timing-function: linear;
}
```

4. Задержка перехода transition-delay

Необязательное свойство, позволяет сделать так, чтобы изменение свойства происходило не моментально, а с некоторой задержкой. Не наследуется.

transition-delay

Значения:	
время	Время задержки перехода указывается в секундах или миллисекундах.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
1     div {
2     transition-delay: .5s;
3     }
```

Плавный переход нескольких свойств

Для элемента можно задать несколько последовательных переходов, перечислив их через запятую. Каждый переход можно оформить своей временной функцией.

```
     div {transition: background 0.3s ease, color 0.2s
 linear;}
```

или

```
     div {
     transition-property: height, width, background-
 color;
     transition-duration: 3s;
     transition-timing-function: ease-in, ease, linear;
     }
```

CSS3 трансформации

CSS3 трансформации изменяют размер, форму и положение элемента на веб-странице с помощью свойства transform. Трансформации преобразовывают элемент, не затрагивая остальные элементы веб-страницы, т.е. другие элементы не сдвигаются относительно него. По умолчанию трансформация происходит относительно центра элемента.

К элементам, которые могут быть трансформированы, относятся элементы с display: block; и display: inline-block;, а также элементы, значение свойства display которых вычисляется как table-row, table-row-group, table-header-group, table-footer-group, table-cell или table-caption. Трансформированным считается элемент с любым установленным значением свойства transform, отличным от none.

Существуют два вида CSS3 трансформаций – **2D** и **3D**. **2D-трансформации** преобразовывают элементы в двухмерном пространстве.

Функции 2D-трансформации transform

Свойство задаёт вид преобразования элемента. Свойство описывается с помощью **функций трансформации**, которые смещают элемент относительно его текущего положения на странице или изменяют его первоначальные размеры и форму. Не наследуется.

Допустимые значения:

matrix() — любое число
translate(), translateX(), translateY() — единицы длины (положительные и отрицательные), %
scale(), scaleX(), scaleY() — любое число
rotate() — угол (deg, grad, rad или turn)
skew(), skewX(), skewY() — угол (deg, grad, rad)

Функция	Описание
none	Значение по умолчанию, означает отсутствие трансформации. Также отменяет трансформацию для элемента из группы трансформируемых элементов.
matrix(a, c, b, d, x, y)	Смещает элементы и задает способ их трансформации, позволяя объединить несколько функций 2D-трансформаций в одной. В качестве трансформации допустимы поворот, масштабирование, наклон и изменение положения. Значение a изменяет масштаб по горизонтали. Значение от 0 до 1 уменьшает элемент, больше 1 — увеличивает. Значение c деформирует (сдвигает) стороны элемента по оси Y, положительное значение — вверх, отрицательное — вниз. Значение b деформирует (сдвигает) стороны элемента по оси X, положительное значение — влево, отрицательное — вправо. Значение d изменяет масштаб по вертикали. Значение меньше 1 уменьшает элемент, больше 1 — увеличивает. Значение x смещает элемент по оси X, положительное — вправо, отрицательное — влево. Значение y смещает элемент по оси Y,

	положительное значение — вниз, отрицательное — вверх.
translate(x,y)	Сдвигает элемент на новое место, перемещая относительно обычного положения вправо и вниз, используя координаты x и y, не затрагивая при этом соседние элементы. Если нужно сдвинуть элемент влево или вверх, то нужно использовать отрицательные значения.
translateX(n)	Сдвигает элемент относительно его обычного положения по оси X.
translateY(n)	Сдвигает элемент относительно его обычного положения по оси Y.
scale(x,y)	Масштабирует элементы, делая их больше или меньше. Значения от 0 до 1 уменьшают элемент. Первое значение масштабирует элемент по ширине, второе — по высоте. Отрицательные значения отображают элемент зеркально.
scaleX(n)	Функция масштабирует элемент по ширине, делая его шире или уже. Если значение больше единицы, элемент становится шире, если значение находится между единицей и нулем, элемент становится уже. Отрицательные значения отображают элемент зеркально по горизонтали.

scaleY(n)	Функция масштабирует элемент по высоте, делая его выше или ниже. Если значение больше единицы, элемент становится ниже, если значение находится между единицей и нулем — ниже. Отрицательные значения отображают элемент зеркально по вертикали.
rotate(угол)	Поворачивает элементы на заданное количество градусов, отрицательные значения от -1deg до -360deg поворачивают элемент против часовой стрелки, положительные — по часовой стрелке. Значение rotate(720deg) поворачивает элемент на два полных оборота.
skew(x-угол,y-угол)	Используется для деформирования (искажения) сторон элемента относительно координатных осей. Если указано одно значение, второе будет определено браузером автоматически.
skewX(угол)	Деформирует стороны элемента относительно оси X.
skewY(угол)	Деформирует стороны элемента относительно оси Y.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
-webkit- 360deg);
-ms-transform: rotate(360deg);
transform: rotate(360deg);
}
```

Точка трансформации transform-origin

Свойство позволяет сместить центр трансформации, относительно которого происходит изменение положения/размера/формы элемента. Значение по умолчанию — center, или 50% 50%. Задаётся только для трансформированных элементов. Не наследуется.

transform-origin

Значения:	
ось X(left, center, right, длина, %) ось Y(top, center, bottom, длина, %)	Пара значений, заданная с помощью ключевых слов, единиц длины или процентов определяет, относительно какой части элемента будет происходить трансформация. Значения больше 100% увеличивают область трансформации элемента.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис

```
div {
-webkit-transform: rotate(45deg);
-ms-transform: rotate(45deg);
transform: rotate(45deg);
-webkit-transform-origin: 20% 40%;
-ms-transform-origin: 20% 40%;
transform-origin: 20% 40%;
}
```

Множественные трансформации

Можно объединить несколько трансформаций одного элемента, перечислив их через пробел в порядке проявления.

```
1    div {transform: scale(1.5) rotate(-10deg);}
```

Примеры CSS анимации

Для примера использования анимации я сделал 7 страниц. В каждом из них приведён пример использования CSS3 анимации.

Первая страница называется ***index.html.*** В нём хранится ссылки для остальных страниц.

Исходный код *index.html*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Animations</title>
    <style>
        ul{
            list-style-type: none;
        }
        a{
            width: 300px;
            text-decoration: none;
            padding: 7px;
            text-align: center;
            background-color: #31f3f1;
            margin-top: 10px;
            display: block;
        }

        a:hover{
            color: red;
            transform: scale(1.1);
            box-shadow: 0 10px 15px -10px rgba(0,0,0,
.5);
        }
    </style>
    <link rel="stylesheet" href="style.css"
type="text/css">
</head>
<body>
<div align="center">
<ul>
    <li><a href="anim1.html">Первая Анимация</a></li>
    <li><a href="anim2.html">Вторая Анимация</a></li>
    <li><a href="anim3.html">Третья Анимация</a></li>
    <li><a href="anim4.html">Четвертая
Анимация</a></li>
    <li><a href="anim5.html">Пятая Анимация</a></li>
```

```
    <li><a href="anim6.html">Шестая Анимация</a></li>
</ul>
</div>
</body>
</html>
```

Вид страницы:

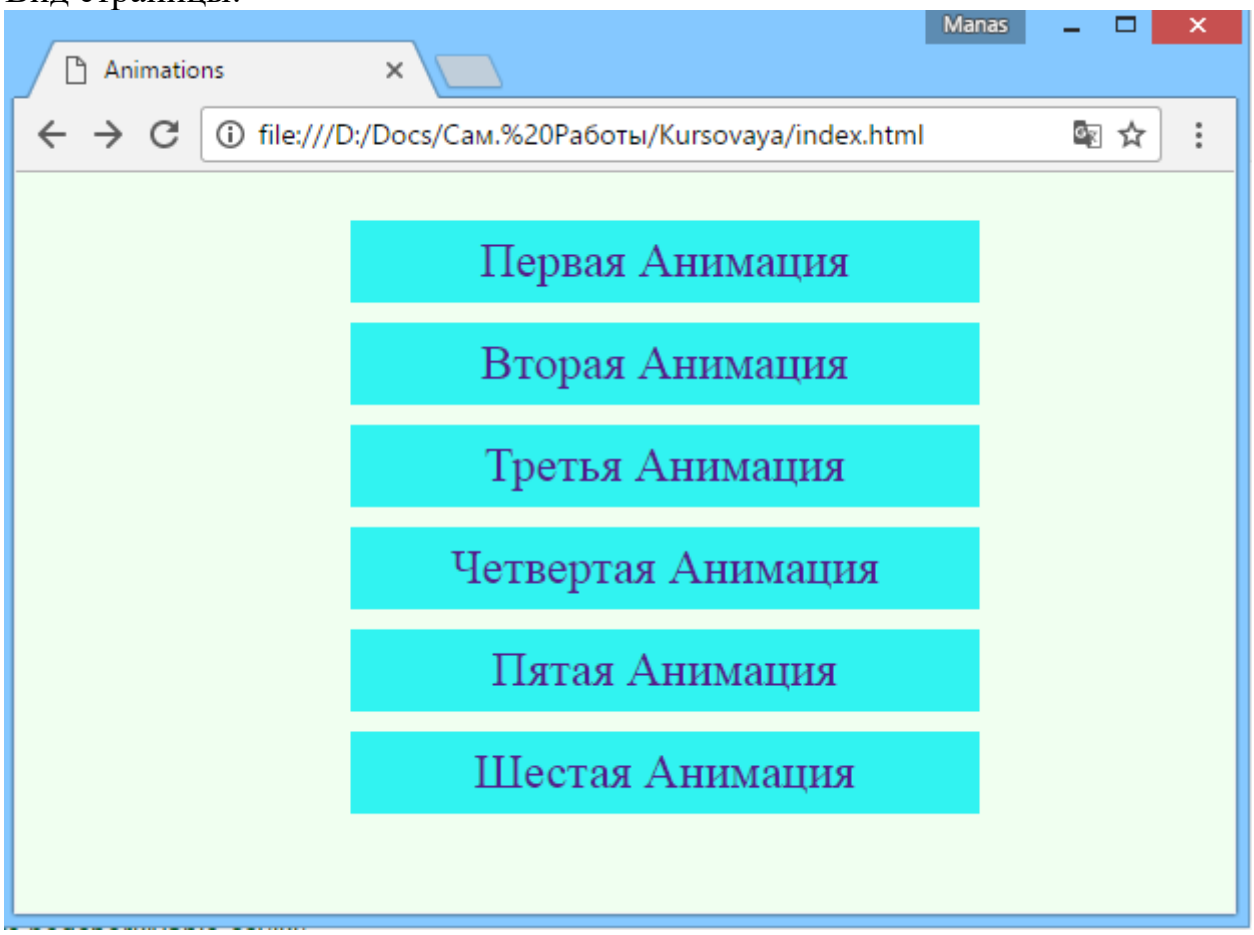


рис.2

В нём можно открыть остальные страницы нажав н соответствующие ссылки. В этой странице я применил CSS переход при наведении на ссылку. Это видно на рис.2.

После нажатия первой ссылки откроется страница anim1.html простая анимация текста с использованием @keyframes рис.3.

HTML код страницы ***anim1.html***:

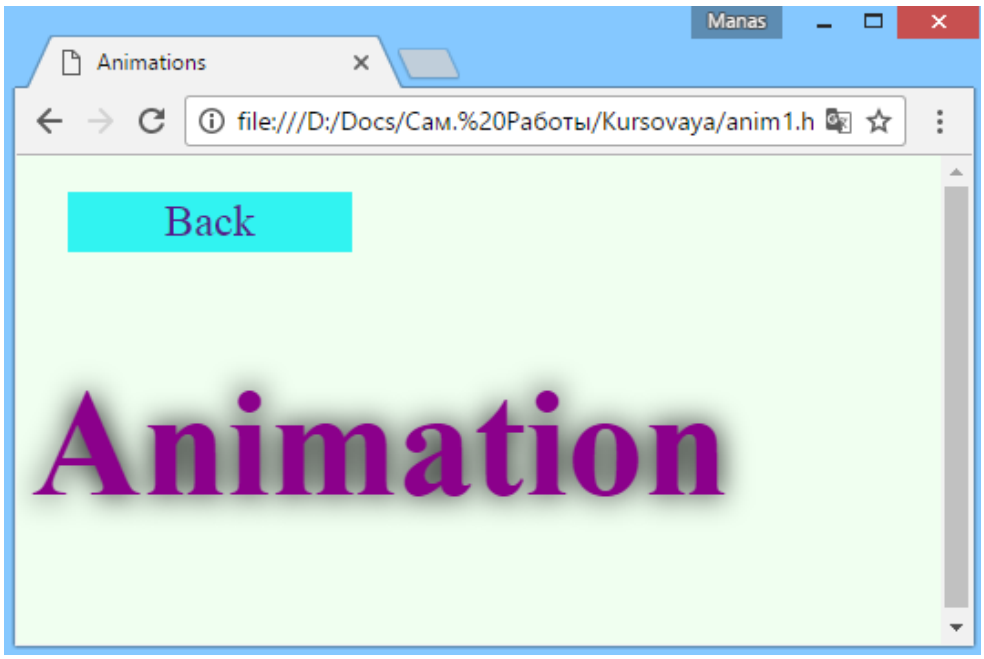
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Animations</title>
    <link rel="stylesheet" href="style.css"
type="text/css">
</head>
<body>
<a href="index.html">Back</a>
```

```
<h1 class="anim">Animation</h1>
</body>
</html>
```

CSS код ***style.css***:

```
body {
  height: 100%;
}
body {
  background: honeydew;
  font-size: 1.5em;
}
a{
  margin: 20px;
  width: 150px;
  text-decoration: none;
  padding: 3px;
  text-align: center;
  background-color: #31f3f1;
  display: block;
}
h1.anim{
  font-size: 3.5em;
  color: darkmagenta;
  animation: shadow  2s infinite ease-in-out;
}
@keyframes shadow {
from {  text-shadow: 0 0 3px black;}
50% {  text-shadow: 0 0 30px black;}
to {  text-shadow: 0 0 3px black;}
}
```

Вид страницы:



Нажав на ссылку Back мы вернёмся в страницу index.html

После нажатия на вторую ссылку открываем страницу anim2.html.

HTML код страницы ***anim2.html***:

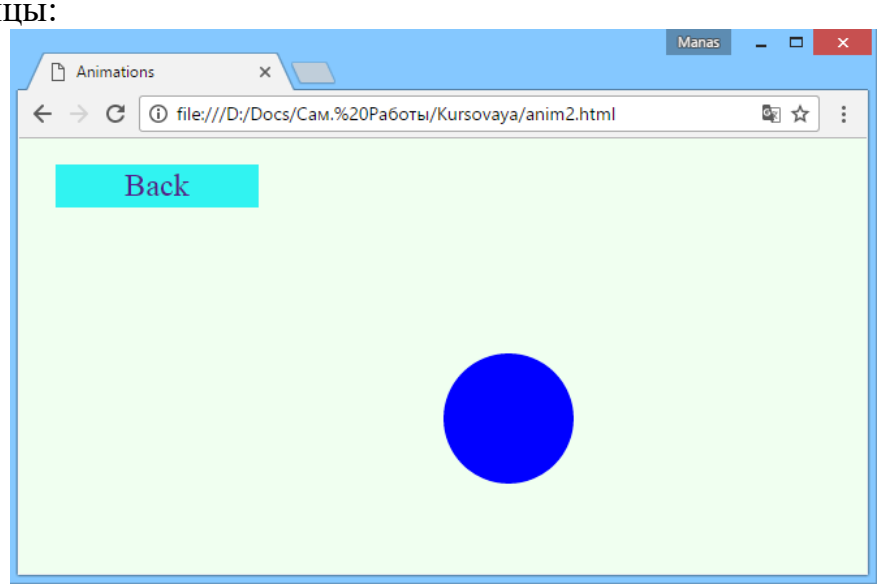
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Animations</title>
    <link rel="stylesheet" href="style2.css"
type="text/css">
</head>
<body>
<a href="index.html">Back</a>
<div class="move"></div>
</body>
</html>
```

CSS код ***style2.css***:body {

```
  height: 100%;
}
body {
  background: honeydew;
  font-size: 1.5em;
}
a{
  margin: 20px;
  width: 150px;
  text-decoration: none;
  padding: 3px;
```

```
  text-align: center;
  background-color: #31f3f1;
  display: block;
}
div.move{
    background-color: #00f;
    position: relative;
    left: 50%;
    width: 100px;
    height: 100px;
    border-radius: 50%;
    animation-timing-function: ease-in-out;
    animation-name: moves;
    animation-duration: 2s;
    animation-delay: 1s;
    animation-iteration-count: infinite;
}
@keyframes moves {
from, to {
top: 0;
left: 50%;
}
25%, 75% {
top: 100px;
}
50% {
top: 50px;
}}
```

Вид страницы:



Нажав на ссылку Back мы вернёмся в страницу index.html

После нажатия на третью ссылку открываем страницу anim3.html.

HTML код страницы ***anim3.html***:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Animations</title>
    <link rel="stylesheet" href="style3.css"
type="text/css">
</head>
<body>
<a href="index.html">Back</a>
<div></div>
<div class="transition"></div>
</body>
</html>
```

Код style3.css:

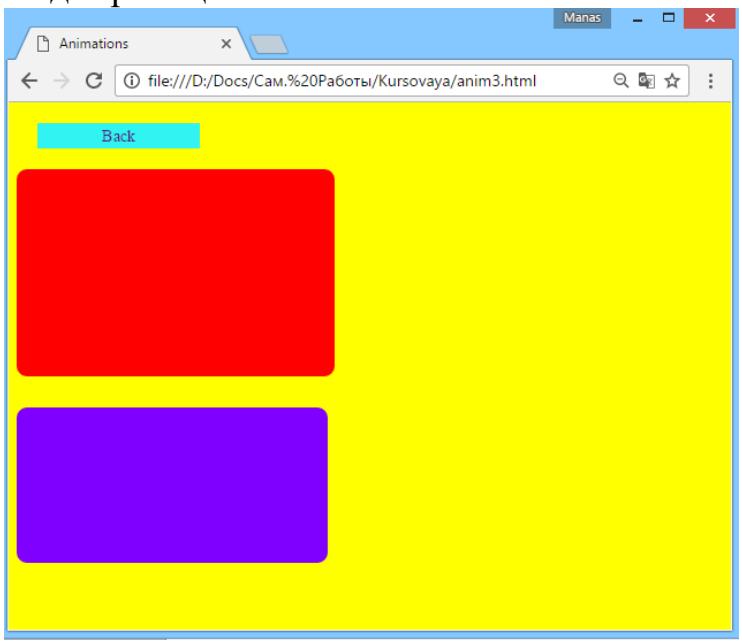
```
body{
  background-color: yellow;
}
a{
  margin: 20px;
  width: 150px;
  text-decoration: none;
  padding: 3px;
  text-align: center;
  background-color: #31f3f1;
  display: block;
}
div{
  background-color: red;
  width: 200px;
  height: 200px;
  border-radius: 10px;
  transition-duration: 2s;
  transition-property: width;
  transition-timing-function: linear;
  transition-delay: .5s;
}
div:hover{
  width: 400px;
}
div.transition {
  margin-top: 30px;

  height: 200px;
```

```
  width: 200px;
  background-color: blue;
transition-property: height, width, background-color;
transition-duration: 1s;
transition-timing-function: ease-in-out;;
}

div.transition:hover{
  height: 100px;
  width: 400px;
  background-color: magenta;
}
```

Вид страницы:



Нажав на ссылку Back мы вернёмся в страницу index.html

После нажатия на четвертую ссылку открываем страницу anim4.html. Здесь показан примеры для использования ***animation-direction.***

HTML код страницы ***anim4.html***:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Animations</title>
    <link rel="stylesheet" href="style4.css"
type="text/css">
</head>
<body>
<a href="index.html">Back</a>
<div class="container">
  <kbd>animation-direction: alternate;</kbd>
<ul class="alternate">
```

```
  <li></li>   <li></li>  <li></li>  <li></li>
<li></li>
</ul>
  <kbd>animation-direction: alternate-reverse;</kbd>
<ul class="alternate-reverse">
  <li></li>  <li></li>  <li></li>  <li></li>  <li></li>
</ul>
  <kbd>animation-direction: normal;</kbd>
<ul class="normal">
  <li></li>  <li></li>  <li></li>  <li></li>  <li></li>
</ul>
  <kbd>animation-direction: reverse;</kbd>
<ul class="reverse">
  <li></li>  <li></li>  <li></li>  <li></li>  <li></li>
</ul>
</div>
</body>
</html>
```

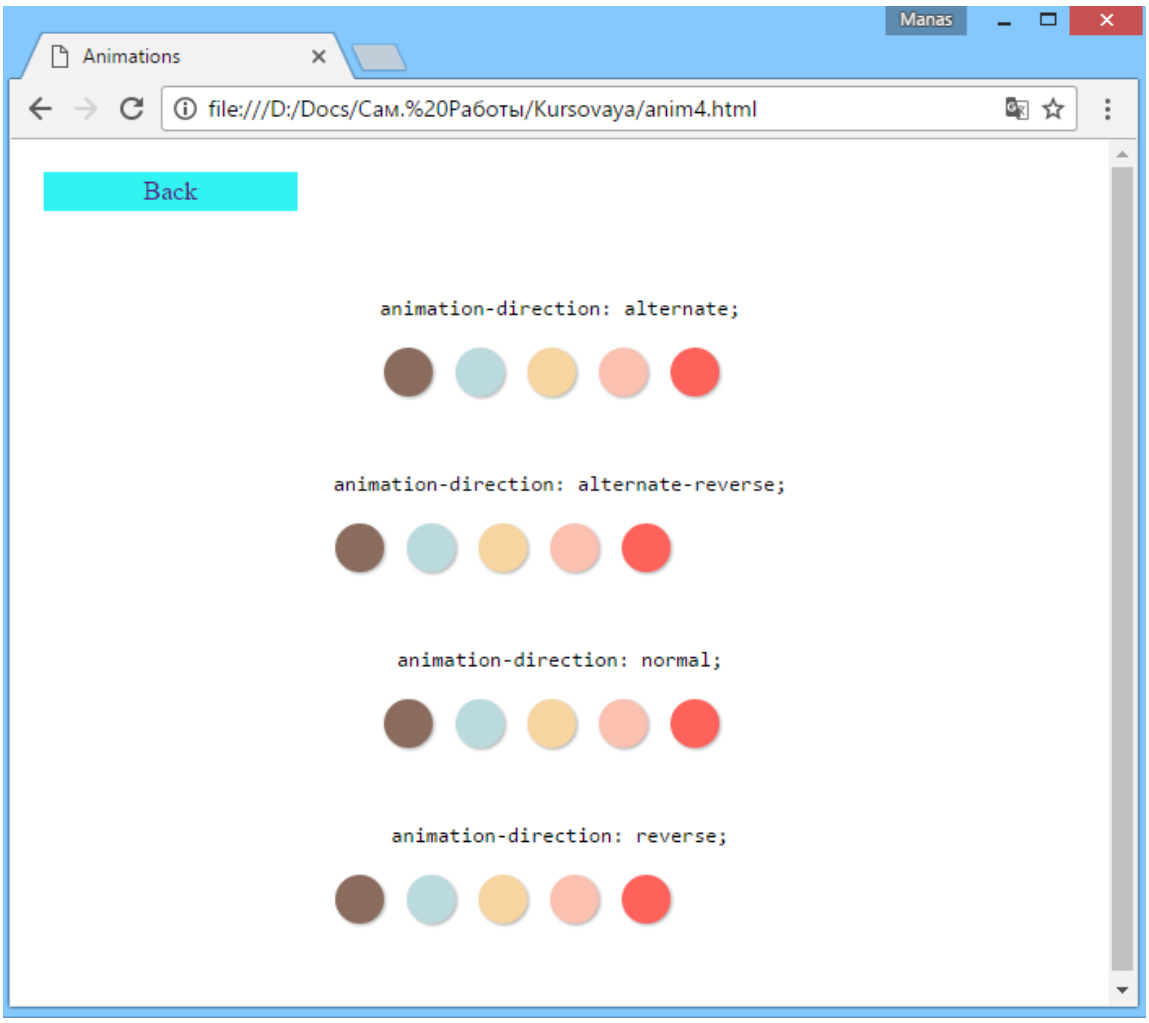
CSS код style4.css :

```
body {
  margin: 0;
  text-align: center;

}
a{
  margin: 20px;
  width: 150px;
  text-decoration: none;
  padding: 3px;
  text-align: center;
  background-color: #31f3f1;
  display: block;
}
.container {
  width: 660px;
  margin: 50px auto;
}
ul {
  padding: 0;
  list-style: none;
  margin-bottom: 40px;
}
li {
  display: inline-block;
```

```
  width: 30px;
  height: 30px;
  border-radius: 50%;
  margin-right: 10px;
  box-shadow: 1px 1px 2px rgba(0,0,0,.3);
}
li:nth-child(1) {background:#8A6C5E}
li:nth-child(2) {background:#BBDADE}
li:nth-child(3) {background:#F6D5A1}
li:nth-child(4) {background:#FAC0B0}
li:nth-child(5) {background:#FF615B}
.alternate li {
  animation: shake .7s linear alternate infinite;
}
.alternate-reverse li {
  animation: shake 0.7s linear alternate-reverse
infinite;
}
.normal li {
  animation: shake 0.7s linear normal infinite;
}
.reverse li {
  animation: shake 0.7s linear reverse infinite;
}
@keyframes shake {
  100% {
    transform: translateX(-30px);
  }
```

Вид страницы:



Нажав на ссылку Back мы вернёмся в страницу index.html

После нажатия на четвертую ссылку открываем страницу anim5.html. Здесь показан примеры для использования ***2D Transform.***

HTML код страницы ***anim5.html***:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Animations</title>
    <link rel="stylesheet" href="style5.css"
type="text/css">
</head>
<body>
<a href="index.html">Back</a>
<h1>CSS3 2D Transform примеры (on
<kbd>:hover</kbd>)</h1>
<div class="wrap"><div
class="one"><kbd>translateX(20px)</kbd></div></div>
<div class="wrap"><div class="two"><kbd>translateY(-
20px)</kbd></div></div>
```

```
<div class="wrap"><div class="three"><kbd>translate(-
20px, 20px)</kbd></div></div>
<div class="wrap"><div class="four"><kbd>scaleX(-
1)</kbd></div></div>
<div class="wrap"><div
class="five"><kbd>scaleY(1.2)</kbd></div></div>
<div class="wrap"><div class="six"><kbd>scale(1.2,
0.5)</kbd></div></div>
<div class="wrap"><div
class="seven"><kbd>rotate(720deg)</kbd></div></div>
<div class="wrap"><div
class="eight"><kbd>skewX(20deg)</kbd></div></div>
<div class="wrap"><div class="nine"><kbd>skewY(-
20deg)</kbd></div></div>
<div class="wrap"><div class="ten"><kbd>skew(-20deg,
20deg)</kbd></div></div>
<div class="wrap"><div
class="eleven"><kbd>matrix(1.2,0,0.5,1.2,50,0)</kbd></d
iv></div>
</body>
</html>
```

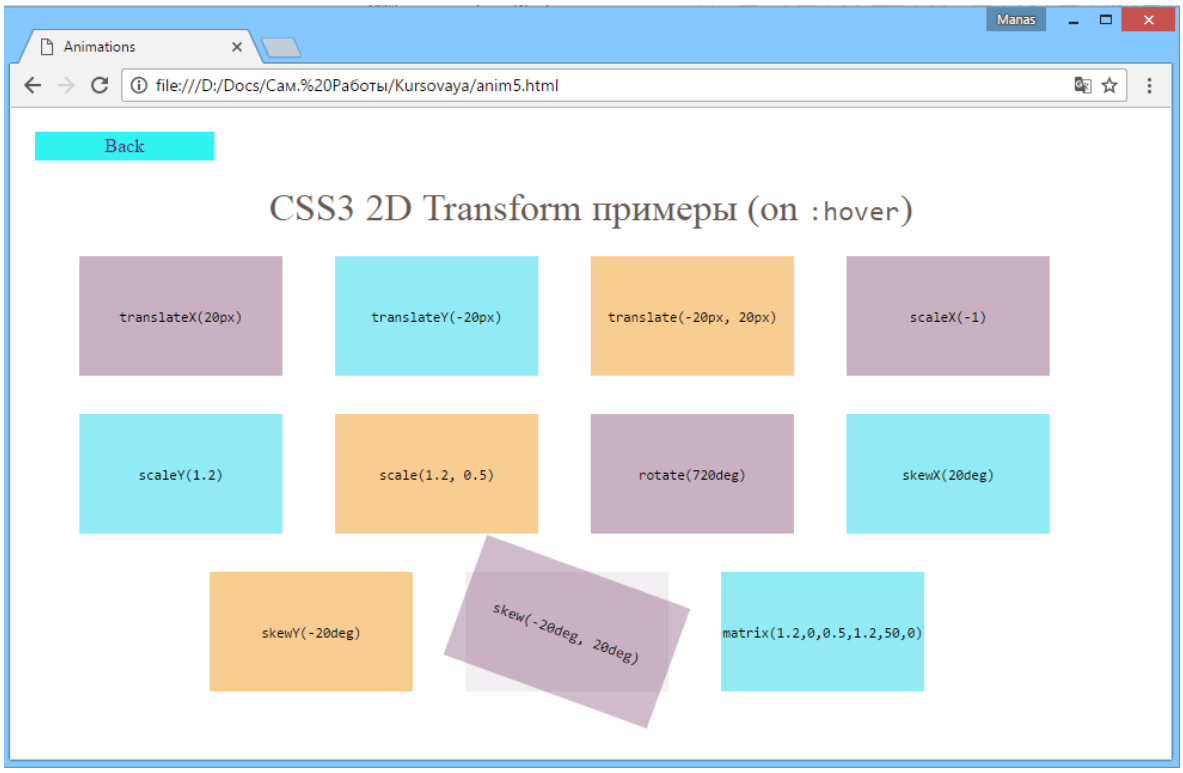
CSS код style5.css:

```
* {box-sizing: border-box;}
body {font-family: 'Playfair Display', serif; margin:
0;text-align: center}
h1 {font-weight: normal;color: #6A5953}
kbd {font-size: 0.9em;display:inline-block;line-
height:1.1;}
.wrap {
  display: inline-block;
  margin: 0 40px 2em 0;
  background: rgba(228, 225, 228, .5);
}
a{
  margin: 20px;
  width: 150px;
  text-decoration: none;
  padding: 3px;
  text-align: center;
  background-color: #31f3f1;
  display: block;
}
div {
  width: 170px;
  height: 100px;
```

```
  line-height: 100px;
  margin: 0 auto;
  transition: all 0.5s ease-in-out;
}
.one, .four, .seven, .ten {
  background: rgba(135, 86, 120, .4);
}
.one:hover {
  transform: translateX(20px);
}
.two, .five, .eight, .eleven {
  background: rgba(0, 228, 246, .4);
}
.two:hover {
  transform: translateY(-20px);
}
.three, .six, .nine {
  background: rgba(255, 153, 0, .4);
}
.three:hover {
  transform: translate(-20px, 20px);
}
.four:hover {
  transform: scaleX(-1);
}
.five:hover {
  transform: scaleY(1.2);
}
.six:hover {
  transform: scale(1.2, 0.5);
}
.seven:hover {
  transform: rotate(720deg);
}
.eight:hover {
  transform: skewX(20deg);
}
.nine:hover {
  transform: skewY(-20deg);
}
.ten:hover {
  transform: skew(-20deg, 20deg);
}
.eleven:hover {
  transform: matrix(1.2,0,0.5,1.2,50,0);
```

}

Вид страницы



Нажав на ссылку Back мы вернёмся в страницу index.html

После нажатия на четвертую ссылку открываем страницу anim6.html. Здесь показан примеры для использования ***Transform-origin*** **.**

HTML код страницы ***anim6.html***:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Animations</title>
    <style>

    </style>
    <link rel="stylesheet" href="style6.css"
type="text/css">
</head>
<body>
<a href="index.html">Back</a>
<div class="container">
  <div class="wrap">
    <div class="element element-1">
      <kbd>50% 50%</kbd>
    </div>
  </div>
  <div class="wrap">
```

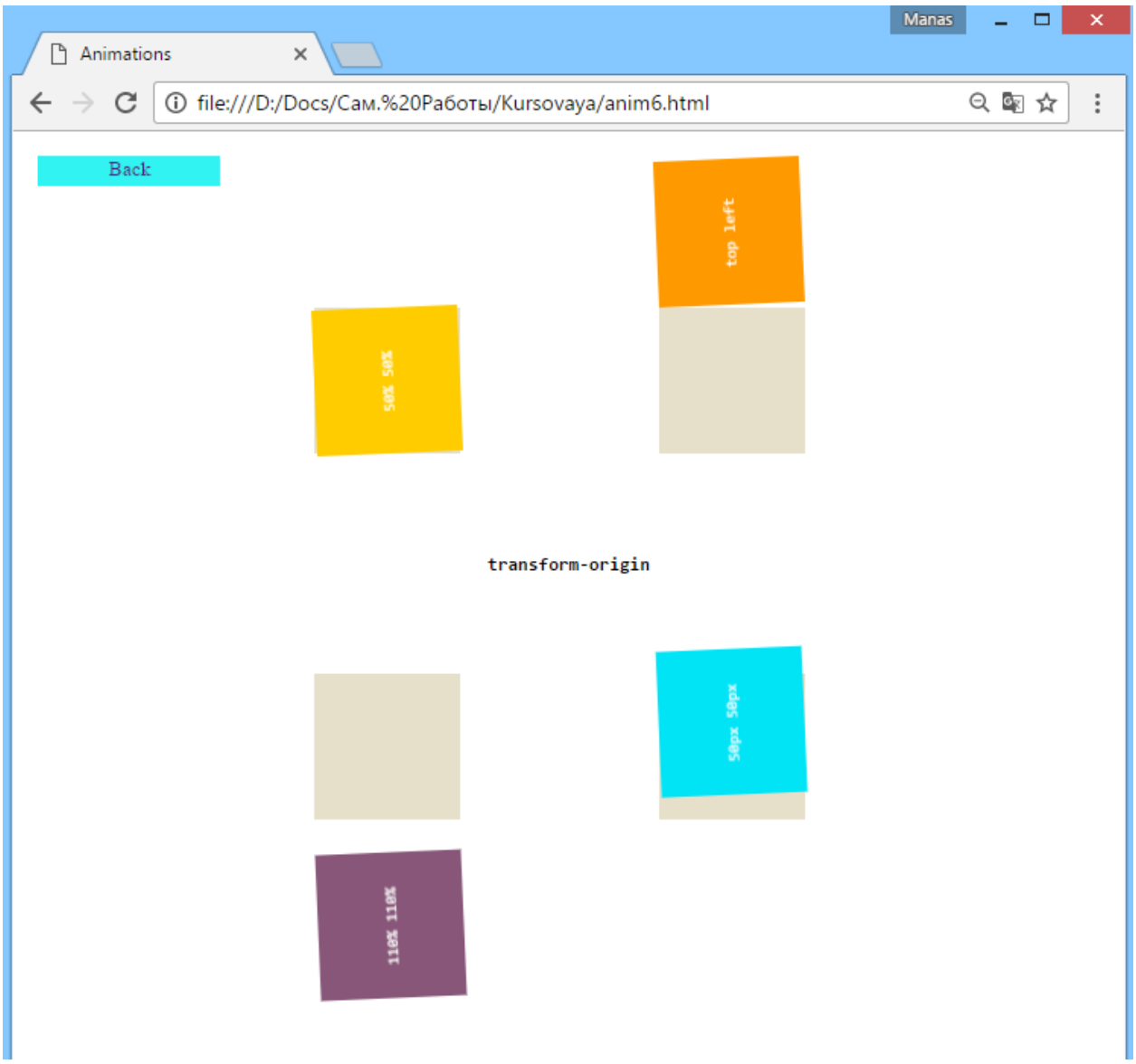
```
    <div class="element element-2">
      <kbd>top left</kbd>
    </div>
  </div>
  <h3><kbd>transform-origin</kbd></h3>
  <div class="wrap">
    <div class="element element-3">
      <kbd>110% 110%</kbd>
    </div>
  </div>
  <div class="wrap">
    <div class="element element-4">
      <kbd>50px 50px</kbd>
    </div>
  </div>
</div>
</body>
</html>
```

CSS код style6.css:

```
*{box-sizing:border-box}
body {margin: 0;overflow-x:hidden}
.container {
  margin: 0 auto;
  max-width: 580px;
}
a{
  margin: 20px;
  width: 150px;
  text-decoration: none;
  padding: 3px;
  text-align: center;
  background-color: #31f3f1;
  display: block;
}
.wrap {
  width: 120px;
  height: 120px;
  background: #E6E0CA;
  display: inline-block;
  margin: 80px;
}
h3{margin:0;text-align:center}
@-webkit-keyframes rotate {
  100% {-webkit-transform: rotate(360deg);
```

```
  -ms-transform: rotate(360deg);
    transform: rotate(360deg);}
}
@keyframes rotate {
  100% {-webkit-transform: rotate(360deg);
  -ms-transform: rotate(360deg);
    transform: rotate(360deg);}
}
.element {
  width: 120px;
  height: 120px;
  color: white;
  text-align: center;
  line-height: 120px;
  -webkit-animation: rotate 6s linear infinite;
  animation: rotate 6s linear infinite;
}
.element-1 {
  background: #FFCC00;
}
.element-2 {
  background: #FF9900;
  -webkit-transform-origin: top left;
  -ms-transform-origin: top left;
  transform-origin: top left;
}
.element-3 {
  background: #875678;
  -webkit-transform-origin: 110% 110%;
  -ms-transform-origin: 110% 110%;
  transform-origin: 110% 110%;
}
.element-4 {
  background: #00E4F6;
  -webkit-transform-origin: 50px 50px;
  -ms-transform-origin: 50px 50px;
  transform-origin: 50px 50px;
}
```

Вид страницы:


Manas
Animations
file:///D:/Docs/Сам.%20Работы/Kursovaya/anim6.html
Back
top left
50% 50%
transform-origin
50px 50px
110% 110%

Заключение

В этой курсовой работе были рассмотрены новые возможности CSS3. С помощью каскадных стилей стали возможны анимация элементов страницы баз использования языка JavaScript. Теперь с новыми возможностями CSS3 веб-страницы стали более красивыми, удобными и лёгкими в понимании.

Анимации являются одним из лучших инструментов для создания интерактивного сайта, разработки рекламы и т.д. С новой версией CSS анимация стала использоваться шире в интернет пространстве.

Для написания исходного кода были использованы инструменты: редактор Sublime Text3 и веб браузер Google Chrome.

Использование принципов, описанных выше позволит веб-разработчикам создавать более качественные и приятно выглядящие анимации. Если анимация учитывает физический вес объекта, выделяет изменения, в ней используются вторичные движения, а ее время исполнения скрупулёзно рассчитано — то это значительно улучшит общее впечатление пользователей от страницы.

Список использованной литературы

1. Кристофер Шмитт “CSS. Рецепты программирования” 3-е изд. Совместный проект издательства «Русская редакция» и издательства «БХВ-Петербург» 647 стр.
2. Зои Джилленуотер “Сила CSS3. Освой новейший стандарт веб-разработок!” — СПб.: Питер, 2012. — 304 с.: ил
3. Секреты CSS. Идеальные решения ежедневных задач, Леа Веру - 2016 год
4. http://codepen.io/
5. https://html5book.ru/