Лабораторная работа № 12

Тема работы: «Создание анимации средствами CSS»

1. Цель работы

Формирование умений создания анимации средствами CSS

2. Задание

Создать эффекты и анимации для разработанного макета используя средства CSS.

3. Оснащение работы

ПК, текстовый редактор, браузер.

4. Основные теоретические сведения

С помощью **CSS3 анимации** можно создавать эффекты, которые раньше воспроизводились с помощью скрипта и с использованием графических редакторов.

Анимация применяется к любым html-элементам, а также к псевдоэлементам **:before** и **:after**.

CSS transitions и **CSS animations** позволяют реализовать анимацию средствами CSS, без привлечения JavaScript.

CSS анимации

CSS анимации позволяют анимировать переходы от одних конфигурации CSS стилей к другим. CSS анимации состоят из двух компонентов: стилевое описание анимации и набор ключей (@keyframes) определяющих начальное, конечное и возможно промежуточное состояние анимируемых стилей.

Пример стилевого описания элемента:

```
h1 {
font-size: 3.5em;
color: darkmagenta;
-webkit-animation: shadow  2s infinite ease-in-out;
animation: shadow  2s infinite ease-in-out;
}
```

Правило @keyframes

```
@keyframes shadow {
from {text-shadow: 0 0 3px black;}
25% {text-shadow: 0 0 30px black;}
50% {text-shadow: 0 0 300px black;}
75% {text-shadow: 0 0 100px black;}
to {text-shadow: 0 0 3px black;}
}
```

Сначала идет объявление @keyframes, которое затем используется в свойстве animation элемента.

Правило @keyframes позволяет создавать анимацию с помощью ключевых кадров — состояний объекта в определенный момент времени.

Синтаксис:

```
@keyframes имя анимации {ключевые кадры {css-стили;}}
```

Ключевые кадры анимации 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 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.

Синтаксис:

```
@-webkit-keyframes shadow {
from {text-shadow: 0 0 3px black;}
50% {text-shadow: 0 0 30px black;}
to {text-shadow: 0 0 3px black;}
}
@keyframes shadow {
from {text-shadow: 0 0 3px black;}
50% {text-shadow: 0 0 30px black;}
to {text-shadow: 0 0 3px black;}
}
```

Также можно комбинировать ключевые слова и процентные пункты. Если кадры имеют одинаковые свойства и значения, их можно объединить в одно объявление:

```
@keyframes move {
from,
to {
top: 0;
left: 0;
}
25%,
75% {top: 100%;}
50% {top: 50%;}
}
```

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

```
h1 {
font-size: 3.5em;
color: darkmagenta;
-webkit-animation: shadow  2s infinite ease-in-out;
animation: shadow  2s infinite ease-in-out;
}
```


Чтобы создать CSS анимацию, необходимо добавить в стиль элемента, который нужно анимировать, свойство animation *или его под-свойства. Это позволит настроить ускорение и продолжительность анимации, а также другие детали о том, как анимация должна протекать.*

Внешний вид анимации настраивается с помощью @keyframes .

Свойство animation имеет следующие свойства и под-свойства:

animation-delay - настраивает задержку между временем загрузки элемента и временем начала анимации;

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;}.

animation-direction - дает возможность при каждом повторе анимации идти по альтернативному пути, либо сбросить все значения и повторить анимацию;

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;}.

animation-duration - настраивает время в течение которого должен пройти один цикл анимации;

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

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

animation-iteration-count - настраивает количество повторении анимации. Можно использовать значание **infinite** для бесконечного повторения анимации;

animation-iteration-count	
Значения:	
Число	С помощью целого числа задается количество повторов анимации. Значение по умолчанию 1.
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;}.

animation-name - определяет имя @keyframes настраивающего кадры анимации;

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: div {animation-name: mymove;}

animation-play-state - позволяет приостановить и возобновить анимацию;

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;}.

animation-timing-function - настраивает ускорение анимации;

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,	Позволяет вручную установить значения от 0 до 1.
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.

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.

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

animation-fill-mode - настраивает значения используемые анимацией до и после исполнения.

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;}.

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

```
animation: animation-name animation-duration animation-timing-function anima-
tion-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.

После того, как настроены временные свойства(продолжительность, ускорение) анимации, необходимо определить внешний вид анимации. Это делается с помощью двух и более ключей после @keyframes. Каждый ключ описывает где должен находиться анимированный элемент в данный момент.

В то время как временные характеристики (продолжительность анимации) указываются в стилях для анимируемого элемента, ключи используют percentage чтобы определить стадию протекания анимации. 0% означает начало анимации, а 100% конец. Так как эти значения очень важны, то для них придумали специальные слова: from и to.

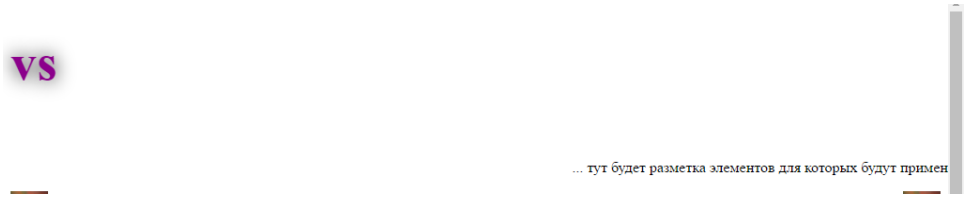
Также можно добавить ключи, характеризующие промежуточное состояние анимации.

Пример анимирует скольжение текста в элементе <p> от правого края окна браузера:

```
p {
  animation-duration: 10s;
  animation-name: slidein;
  animation-iteration-count:infinite
}

@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
}
```



Обратите внимание, что анимация может сделать страницу шире чем окно браузера. Этого можно избежать, поместив элемент который будет анимироваться в контейнер и установить ему свойство overflow:hidden.

В стиле для элемента <p> с помощью свойства animation-duration указано, что исполнение анимации от начала до конца должно занять 10 с , и что имя для @keyframes, описывающей саму анимацию это "slidein".

В элемент <p> можно добавлять и другие пользовательские стили чтобы как-то украсить его, однако здесь мы хотели продемонстрировать только эффект анимации.

Ключи определяются с помощью правила @keyframes. В данном случае мы имеем только два ключа. Первый при 0% анимации (from). Здесь мы придаем элементу левый отступ в 100% и ширину в 300% (в три раза больше ширины

родительского элемента). Это становится причиной того, что при первом кадре анимации заголовок <p> находится за пределами правого края окна браузера .

Второй ключ (to) определяет конец анимации, т.е (100%). Левый отступ устанавливается равным 0, а ширина 100%. Все выглядит, будто заголовок <p> приплывает к левому краю окна браузера.

Добавим другие ключи в предыдущий пример, чтобы размер шрифта заголовка возрастал на время по мере продвижения влево, а потом возвращался к первоначальному значению. Это легко реализовать с помощью следующего ключа:

```
75% {
  font-size: 300%;
  margin-left: 25%;
  width: 150%;
}
```

Текст при каждом повторении снова "запрыгивает" за край окна браузера.Сделаем, чтобы текст двигался влево и вправо. Этого легко достичь посредством установки свойству animation-direction *значения* alternate*:*

```
p {
  animation-duration: 3s;
  animation-name: slidein;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

Пример:

```
@keyframes rotate {
        0% {
        transform: rotate(0deg);
    }
        50% {
        transform: rotate(180deg);
    }
        100% {
        transform: rotate(360deg);
    }
}

.wood-wheel {
    animation-name: rotate;
    animation-duration: 2s;
}
```

Пример:

```
@keyframes go-left-right {
    from {
      left: 0px;
    }
    to {
      left: calc(100% - 50px);
    }
  }
.progress {
    animation: go-left-right 3s infinite alternate;
    position: relative;
    border: 2px solid green;
    width: 50px;
```

```
    height: 20px;
    background: lime;
  }
```

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;}
```

Можно выделить три преимущества CSS анимации перед традиционными способами:

1. Простота использования для простых анимаций. *Вы можете создать анимацию не зная JavaScript.*
2. Анимации будут хорошо работать даже при умеренных нагрузках системы.
3. Позволяет браузеру контролировать последовательность анимации, тем самым оптимизируя производительность и эффективность браузера. *Например, уменьшая частоту обновления кадров анимации в непросматриваемых в данный момент вкладках.*

CSS transitions

Transition – универсальное свойство, которое определяет эффект перехода между двумя состояниями элемента, они могут быть установлены ко всем элементам, а также с помощью псевдоэлемента :hover или :active, а также динамически через JavaScript. *Наиболее часто используются для стилизации ссылок и кнопок в состоянии :hover или :focus.*

Свойство **transition** имеет следующие под-свойства:

transition-delay - устанавливает время ожидания перед запуском эффекта анимации перехода. Не наследуется *(см. другие св-ва)*;

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.

Пример:

```
div {
-webkit-transition-delay: .5s;
transition-delay: .5s;

}
```

Пример:

```
<div class="wrap">
  <h1>Наведите на блок</h1>
  <kbd>transition-delay</kbd>
  <div class="container">
  <span>0s</span>
  <div class="box1 box"></div>
```

```
  </div>
  <div class="container">
    <span>0.5s</span>
  <div class="box2 box"></div>
  </div>
  <div class="container">
    <span>1s</span>
  <div class="box3 box"></div>
  </div>
</div>

body {font-family: 'Playfair Display', serif; margin: 0;}
.wrap {
  margin: 20px 0 0;
  text-align: center;
}
.container {display:inline-block}
h1 {
  color: #3A262F;
  font-weight:normal;
}
kbd{display:block;margin-bottom:15px;}
.box {
  border-radius: 8px;
  margin: 20px;
  width: 60px;
  height: 60px;
}
.wrap:hover .box {
  transform: scale(1.5);
}

.box1 {
  background:#BED3E4;
  transition: 1s linear;
}
.box2 {
  background:#BCB4D9;
  transition: 1s linear .5s;
}
.box3 {
  background:#EBC0AD;
  transition: 1s linear 1s;
}
```

transition-duration - задаёт время в секундах или миллисекундах, сколько должна длиться анимация перехода до её завершения;

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.

Пример:

```
div {
-webkit-transition-duration: 5s;
transition-duration: 5s;
}
```

Пример:

```
<div class="wrap">
  <h1>Наведите на блок</h1>
  <kbd>transition-duration</kbd>
  <div class="container">
    <span>0.5s</span>
    <div class="box1 box"></div>
  </div>
   <div class="container">
    <span>1s</span>
    <div class="box2 box"></div>
  </div>
   <div class="container">
    <span>2s</span>
    <div class="box3 box"></div>
  </div>
</div>


body {font-family: 'Playfair Display', serif; margin: 0;}
.wrap {text-align: center;}
.container {
  display: inline-block;
}
h1 {
  color: #3A262F;
  font-weight:normal;
  text-align: center;
}
kbd{display:block;text-align:center;margin-bottom:20px;}
.box {
  height: 40px;
  width: 40px;
  border-radius: 50%;
  margin: 20px 50px 0;
}
.wrap:hover .box {
  -webkit-transform: scale(2);
  -ms-ransform: scale(2);
  transform: scale(2);
}
.box1 {
  background: #FFD592;
  transition: .5s;
}
.box2 {
  background: #F9A88B;
  transition: 1s;
}
.box3 {
  background: #B39190;
  transition: 2s;
```

transition-property - устанавливает имя стилевого свойства, значение которого будет отслеживаться для создания эффекта перехода;

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.

Пример:

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

transition-timing-function - устанавливает, насколько быстро должно изменяться значение стилевого свойство для которого применяется эффект перехода. Если определено более одного перехода для элемент, например, цвет фона элемента и его положение, можно использовать разные функции для каждого свойства. Не наследуется.

transition-timing-function	
Значения:	
Ease	Функция по умолчанию, переход начинается медленно, разгоняется быстро и замедляется в конце. Соответствуетcubic-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,	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.

Пример:

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

Пример:

```
<article id="go">
  <a href="#go"><span class="fa fa-power-off"></span>Запустить</a>
  <a href="#reset"><span class="fa fa-refresh"></span>Назад</a>
  <div><kbd>transition-timing-function</kbd></div>
  <p><kbd>linear</kbd></p>
  <p><kbd>ease</kbd></p>
  <p><kbd>ease-in</kbd></p>
  <p><kbd>ease-out</kbd></p>
  <p><kbd>ease-in-out</kbd></p>
  <p><kbd>cubic-bezier</kbd></p>
</article>

body {font-family: 'Playfair Display', serif; margin: 0;line-
height:1;color:#696150;}
article{max-width:660px;margin: 30px auto 0;}
p {
  width: 125px;
  padding: 10px 0;
  text-align: center;
  border-radius: 4px;
  transition-duration: 3s;
}
a {
  text-decoration: none;
  display: inline-block;
  margin-right: 20px;
  color:#696150;
}
.fa {padding-right: 10px}
.fa.fa-power-off {color:#FF9E3B}
.fa.fa-refresh {color:#B7CCCC}
div{margin-top:15px;}
p:nth-of-type(1) {
background: #F06E58;
transition-timing-function: linear;}
p:nth-of-type(2) {
background: #FEA45D;
transition-timing-function: ease;}
p:nth-of-type(3) {
background: #FDF569;
transition-timing-function: ease-in;}
p:nth-of-type(4) {
background: #B9EA7B;
transition-timing-function: ease-out;}
p:nth-of-type(5) {
background: #B9ECFE;
transition-timing-function: ease-in-out;}
p:nth-of-type(6) {
background: #82DDFF;
transition-timing-function: cubic-bezier(0.6, 0.1, 0.15, 0.8);}
```

```
#go:target p{transform:translateX(535px)}
```

Свойство **transition** позволяет одновременно задать значения **transition-property**,**transition-duration**, **transition-timing-function** и **transition-delay**.

Синтаксис

```
transition: <переход> [, <переход> ]*
```

Здесь:

```
<переход> = [ none | <transition-property> ] || <transition-duration>
||
<transition-timing-function> || <transition-delay>
```

none - отменяет эффект перехода.

Для задания всех свойств перехода обычно используют краткую запись свойства **transition**.

Пример:

```
transition: all 0.5s ease-in-out;
```

Если воспользоваться значениями по умолчанию, то запись

```
div {transition: 1s;}
```

будет эквивалентна

```
div {transition: all 1s ease 0s;}
```

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

Пример:

```
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 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. Трансформации не действуют на строчные элементы display: inline.

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.

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.

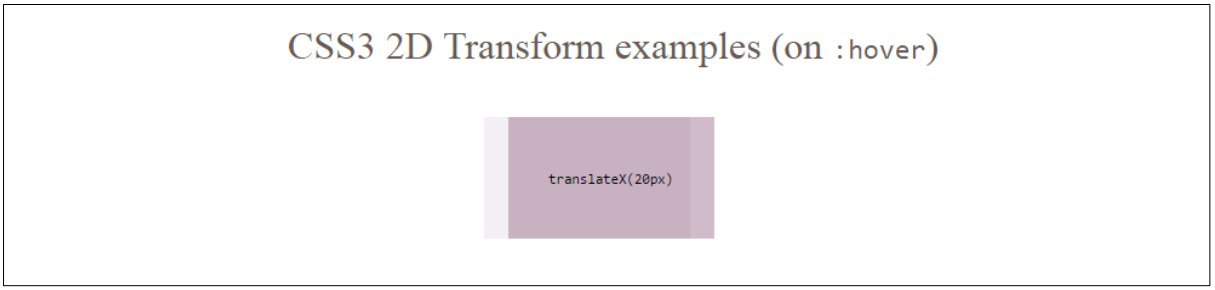
scaleX(n)	Функция масштабирует элемент по ширине, делая его шире или уже. Если значение больше единицы, элемент становится шире, если значение находится между единицей и нулем, элемент становится уже. Отрицательные значения отображают элемент зеркально по
scaleY(n)	Функция масштабирует элемент по высоте, 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.

Пример:

```
<!doctype html>
<head>
        <meta charset="windows-1251" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
        <title>Анимация CSS3</title>
        <link type="text/css" rel="stylesheet" href="style/tanime2.css" me-
dia="all" />
        </head>
<body>
<h1>CSS3 2D Transform examples (on <kbd>:hover</kbd>)</h1>
<br>
<div class="wrap"><div class="one"><kbd>translateX(20px)</kbd></div></div>
</body>
</html>

файл .css
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);}
div{
  width: 170px;
  height: 100px;
  line-height: 100px;
  margin: 0 auto;
  -o-transition: all 0.5s ease-in-out;
```

```
  -moz-transition: all 0.5s ease-in-out;
  -webkit-transition: all 0.5s ease-in-out;
  transition: all 2s ease-in-out;
}
.one {
  background: rgba(135, 86, 120, .4);
}
.one:hover {
  -o-transform: translateX(20px);
  -ms-transform: translateX(20px);
  -moz-transform: translateX(20px);
  -webkit-transform: translateX(20px);
  transform: translateX(20px);
}
```



Пример:

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>transition</title>
  <style>
   #bar {
    top:-5.5em; right:5em; /* Положение */
     padding: .5em; /* Поля */
     margin: 0; /* Отступы */
     position: absolute; /* Абсолютное позиционирование */
     width: 2em; /* Ширина */
     background: #333; /* Цвет фона */
     color: #fff; /* Цвет текста */
     text-align: center; /* Выравнивание по центру */
     /* Переход */
     -webkit-transition: top 1s ease-out 0.5s;
     -moz-transition: top 1s ease-out 0.5s;
     -o-transition: top 1s ease-out 0.5s;
     transition: top 1s ease-out 0.5s;
    }
   #bar:hover { top: 0; }
  </style>
 </head>
 <body>
  <ul id="bar">
   <li>1</li><li>2</li>
   <li>3</li><li>4</li>
   <li>&darr;</li>
  </ul>
 </body>
</html>
```

В данном примере при наведении курсора на стрелку, раскрывается блок с числами.

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	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.

```
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%;
}
```

Пример:

```
div{
  width: 170px;
  height: 100px;
  line-height: 100px;
  margin: 0 auto;
  -o-transition: all 0.5s ease-in-out;
  -moz-transition: all 0.5s ease-in-out;
  -webkit-transition: all 0.5s ease-in-out;
  transition: all 2s ease-in-out;
}
div:hover {
transform: rotate(45deg);
transform-origin: 50% 50%;}
}
```

Множественные трансформации

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

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

Пример:

```
div:hover {
transform: scale(1.5) rotate(-10deg);}
transform-origin: 0% 0%;}
```

Transform-style - определяет, как дочерние элементы должны отображаться в 3D-пространстве. Это свойство должно использоваться совместно с transform.

Синтаксис

```
transform-style: flat | preserve-3d
```

flat - дочерние элементы лежат в той же плоскости, что и их родитель;
preserve-3d - дочерние элементы будут отображаться в 3D-пространстве.

Пример:

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>transform-style</title>
  <style>
   .turn {
    -webkit-transform-style: preserve-3d;
    transform-style: preserve-3d;
   }
   .turn:hover {
    -webkit-transform: rotateY(45deg);
    transform: rotateY(45deg);
   }
  </style>
 </head>
 <body>
  <p><img src="images/igels.png" alt="Ёжик" class="turn">
     <img src="images/igels.png" alt="Ёжик" class="turn"></p>
 </body>
</html>
```

5. Порядок выполнения работы

1. Создать HTML страницу с использованием CSS-анимации, добавив индикатор загрузки, который будет заполняться плавно при наведении.
2. . Создать HTML страницу с использованием CSS-анимации, добавив круг который будут передвигаться по контуру экрана по часовой стрелке, реализуйте плавную смену фона у круга.
3. . Создать HTML страницу с кнопками для навигации, как показано на рисунке:

Добавить эффект анимации для ссылок в соответствии с вариантом.
4. Выполнить задание, соответствующее вашему варианту.

Вариант 1 Создайте мячик, который будет падать сверху вниз, реализуйте ускорение скорости

При наведении на ссылку она должна увеличиваться и сдвигаться вверх, а при нажатии на ссылку она должна уменьшиться по ширине.

Вариант 2 Создайте птицу, которая будет лететь слева направо. Полет птицы реализовать со взмахом крыльев

При наведении на ссылку она должна увеличиваться и сдвигаться вправо, а при нажатии на ссылку она должна уменьшиться по высоте.

Вариант 3 Реализуйте движение машины по проезжей части. При наведении на машину машина останавливается и двери открываются.

При наведении на ссылку она должна увеличиваться и сдвигаться вниз.

Вариант 4 Создайте кофемашину. При наведении на значок кофе из кофемашины в кружку будет наливаться кофе. Реализовать процесс заполнения кружки кофе

При наведении на ссылку она должна отображаться зеркально, при этом необходимо соблюдать очередность: 1-ая и 3-я ссылки зеркально по горизонтали, 2-ая зеркально по вертикали.

Вариант 5 Создайте тучу. При наведении на тучу должен пойти дождик. Дожди должен усиливаться со временем (начинаться медленно – заканчиваться быстро)

При наведении на ссылку она должна увеличиваться и делать один полный поворот по часовой стрелке.

Вариант 6 Создайте двух игроков и футбольный мяч. При наведении на игроков мячик должен лететь к другому игроку.

При наведении на ссылку она должна увеличиваться, при этом ее необходимо деформировать (исказить стороны) на 30^0.

Вариант 7 Создать часы с секундной стрелкой и, используя анимацию CSS, осуществите движение стрелки в обратном порядке. При выведении курсора из области часов стрелка должны останавливаться.

При наведении на ссылку она должна она должна выполнить вращение.

Вариант 8 Создайте коробку. При наведении на коробку коробка должна открываться и появляться надпись «CSS-анимация»

При наведении на ссылку она должна увеличиваться и делать один полный поворот по часовой стрелке относительно верхнего правого угла.

5* Реализуйте плавный полет снежинок

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

_____

7. Контрольные вопросы и задания

1. Перечислите способы создания анимации средствами CSS. Укажите их отличия.

2 Что такое «ключевые кадры»?

3. Укажите формат описания CSS animations.

4. Укажите формат описания CSS transitions.

5. Перечислите под-свойства свойства animation.

6. Перечислите под-свойства свойства transitions.

7. Раскройте суть понятия «множественные трансформации».

8. Рекомендуемая литература

Макфарланд, Д. Новая большая книга CSS / Дэвид Макфарланд. — СПб.: Питер, 2016. — 720с.

Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. — СПб.: Питер, 2016.

Прохоренок, Н.А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А.Прохоренок. – СПб.: БХВ-Петербург, 2010.

Фрейн, Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Бен Фрейн. — СПб.: Питер Пресс, 2017. — 272с.