top

КОМПЬЮТЕРНАЯ АКАДЕМИЯ

HyperText Markup Language

HTML&CSS

Урок 4/1

Анимация в веб-дизайне

Оглавление

Анимация в веб-дизайне.....3
- CSS-переходы. Свойство Transition.....6
- CSS-трансформация. Свойство Transform.....18
- CSS-анимация. Свойство Animation.....40
- Пример предзагрузчика-индикатора для сайта.....49

Домашнее задание.....53
- Задание 1. Реализовать html-страницу с часами.....53
- Задание 2. Реализовать html-страницу со створками.....53
- Задание 3. Реализовать html-страницу с несколькими блоками текст + картинка.....54

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Анимация в веб-дизайне

Анимация — это одна из составляющих частей дизайна сайта. Благодаря анимации можно привлечь внимание пользователя к наиболее важным элементам сайта. Какие элементы на сайте будут в движении, продумывают UX/UI дизайнеры.

- UI — User Interface (пользовательский интерфейс);
- UX — User Experience (опыт пользователя).

UX/UI дизайнеры — это специалисты, которые создают пользовательские интерфейсы, при этом продумывая то, как пользователь будет взаимодействовать с интерфейсом, чтобы получить желаемое, достигнуть цели.

Примеры анимации на сайтах:

- плавно меняющийся фон и тень кнопки, при наведении курсора мыши, на сайте https://www.microsoft.com/en-us/microsoft-365/enterprise;

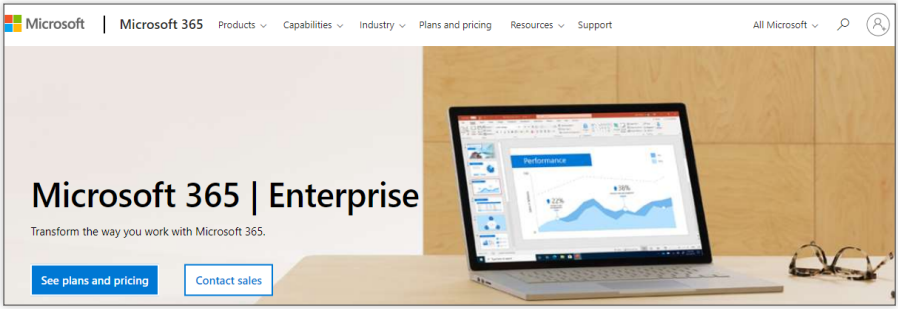

Рисунок 1. Кнопка в спокойном состоянии, без наведения курсора мышки

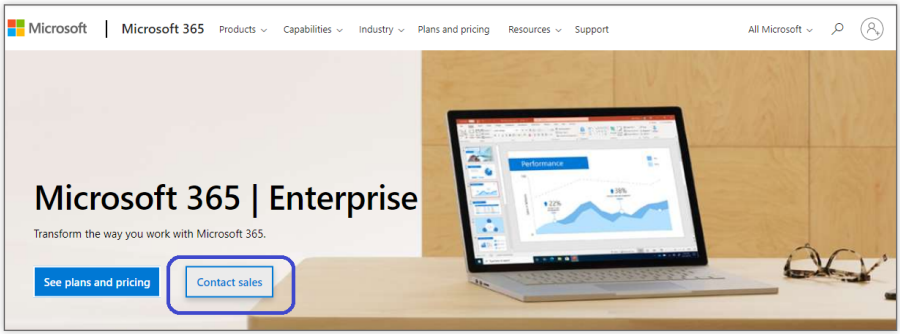

Рисунок 2. При наведении на кнопку «Связаться с отделом продаж», изменяется фон кнопки и добавляется тень

- движущийся фон с обложками кинофильмов, на сайте apple: https://www.apple.com/tv/;

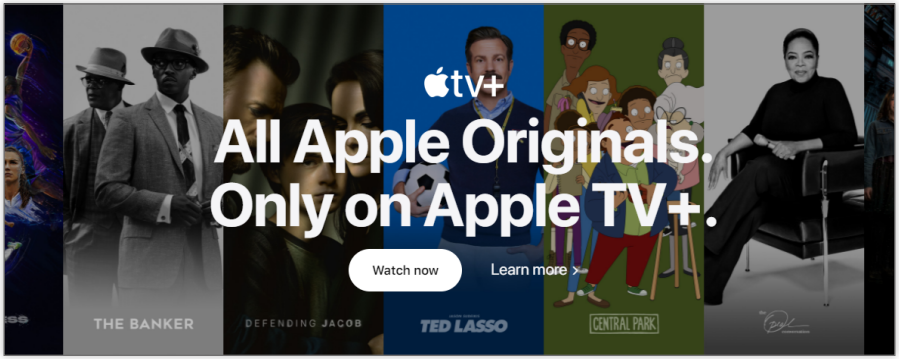

Рисунок 3. Сайт https://www.apple.com/tv/

- плавно меняющийся, при наведении, фон кнопок, текст ссылок на сайте https://dragone.com/en;

Рисунок 4. Сайт https://dragone.com/en

Качественно выполненная анимация, вызывает положительные эмоции у пользователя.

Анимация с помощью свойств CSS3 позволяет создавать простые анимации в браузере, без использования JavaScript. CSS анимации также эффектны и в некоторых случаях их использование более выгодно, по сравнению с анимациями, созданных на основе JavaScript. Несколько лет назад была популярна библиотека jQuery, которая в связке с JavaScript позволяла выполнять различные виды анимаций на странице. По производительности, использование библиотеки jQuery при создании анимации, уступает использованию CSS анимаций. Однако, новые технологии, например библиотека GSAP, на основе JavaScript имеет иногда даже лучшую производительность. Поэтому на данный момент времени, в большинстве случаев, производительность CSS анимаций практически идентична анимациям, созданным на JavaScript. Важным аспектом в выборе инструмента для создания анимации, является возможность управлять конкретными местами в анимации. Для CSS анимаций браузеры предоставляют возможность остановить/перезапустить анимацию, но нет возможности обратиться к какой-то конкретной части анимации, нет возможности изменить направление хода элемента на определённом участке анимации и пр. JavaScript позволяет в полной мере контролировать ход анимации.

CSS анимации подходят для выполнения простых движений элемента и когда не требуется совместимость с устаревшими браузерами.

JavaScript позволяет создавать сложные анимации, с возможностью интерактивного взаимодействия.

Эффект движущихся элементов на сайте, можно создать с помощью свойств CSS3:

- переходов — transition;
- анимации — animation;
- трансформации — transformation. Следует отметить, что трансформация не создаёт анимацию, а только трансформирует (перемещает, вращает, увеличивает) элемент. Трансформации часто используются совместно со свойствами animation и transition.

CSS-переходы. Свойство Transition

CSS-переходы (transition), позволяют изменять CSS-свойствам одно значение на другое, за определённый промежуток времени. Простейший пример подобного поведения — это изменение цвета фона ссылки-кнопки с помощью псевдокласса :hover, при котором моментально происходит смена значений CSS-свойства background-color.

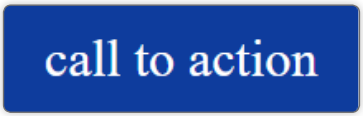

Рисунок 5. Кнопка в спокойном состоянии, без наведения кнопки мыши

Рисунок 6. Цвет фона кнопки, при наведённом курсоре мыши

Создадим эту кнопку (*пример кода во вложении к методпакету, в файле с названием «ButtonCallToAction.html»*).

HTML-код:

```
<a href="#">call to action</a>
```

Неактивная ссылка, начальное состояние:

CSS-код:

```
a{
    text-decoration: none;
    display: inline-block;
    font-size:  2rem;
    background-color: #003fa3;
    border-radius: 5px;
    color: #ffffff;
    padding: 15px 25px;
}
```

Рисунок 7. Неактивная ссылка, начальное состояние

Ссылка в состоянии :hover, конечное состояние:

CSS-код:

```
a:hover{
    background-color: #012763;
}
```

Рисунок 8. Ссылка в состоянии :hover, конечное состояние

Дополним первое правило свойством transition:

CSS-код:

```
a{
    text-decoration: none;
    display: inline-block;
    font-size:  2rem;
    background-color: #003fa3;
    border-radius: 5px;
    color: #ffffff;
    padding: 15px 25px;
    transition: all 0.8s ease 0s;
}
```

transition: all 0.8s ease 0s; — сокращенное свойство, определяющее переход. Первое значение определяет список свойств, чьи переходы должны анимироваться. Второе определяет время перехода. Третье значение определяет временную функцию. Четвертое определяет время задержки до начала анимации. Обратите внимание, что ключевое слово all затронуло только цвет фона, потому что для состояния :hover (при наведении курсора мыши), указано только свойство background-color, т.е. изменяется только цвет фона.

После добавления сокращенного свойства transition, цвет фона ссылки-кнопки будет плавно переходить

из первого (начального) состояния во второе (конечное, состояние :hover), в течение 0,8 секунд.

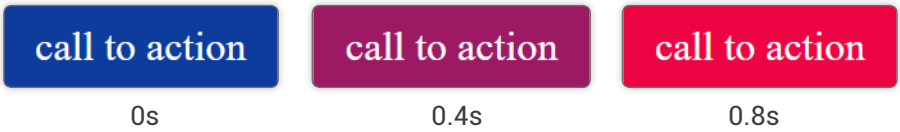

Рисунок 9. Изменение цвета фона во времени

Стоит отметить, что переход применяется к исходному, невыделенному элементу. Это делается для того, чтобы при других состояниях ссылки, возможно было добавлять другие стили и также выполнять для них переход. CSS-свойство transition необходимо применять для элемента в том состоянии, из которого будет осуществляться переход.

Объявить переход можно с помощью сокращенного свойства transition (как указано в примере выше) или с помощью 4-х отдельный свойств: transition-property, transition-duration, transition-timing-function, transition-delay. В этом случае первое правило будет выглядеть следующим образом:

CSS-код:

```
a{
    text-decoration: none;
    display: inline-block;
    font-size:  2rem;
    background-color: #003fa3;
    border-radius: 5px;
    color: #ffffff;
    padding: 15px 25px;
    transition-property: background-color;
```

```
    transition-duration: 0.8s;
    transition-timing-function:  ease;
    transition-delay: 0s;
}
```

- transition-property — определяет список изменяемых свойств. Можно указать одно свойство или список свойств, которые должны изменяться, через запятую.

Добавим в правило, для выделяемого элемента, изменение свойств цвета текста и размера шрифта:

CSS-код:

```
a:hover{
    background-color: #012763;
    font-size:  2.1rem;
    color: #ffff00;
}
```

Для того чтобы плавно изменялись только свойства цвета фона и размер шрифта, в значениях transition-property необходимо указать именно эти свойства, через запятую:

```
transition-property: background-color, font-size;
```

Значение all означает, что все измеряемые свойства элемента будут изменяться. Значение none означает, что никакое свойство не будет изменяться.

Изменению подлежат те CSS свойства, которые изменяют внешний вид элемента или его расположение в HTML документе. К таким свойствам относятся: width, height, left, top, right, bottom, color, opacity, background-color, margin-

left, border и т.д (с полным списком изменяемых свойств можно ознакомиться на сайте developer.mozilla.org).

Стоит отметить, что переход можно применить к той паре значений свойств, у которых можно определить измеряемую среднюю точку. Например, для свойства width, можно применить переход от значения 400px до 800px, так есть некая средняя точка 600px. А для элемента, у которого изменяется свойство display: none на display: block, невозможно применить переход, так как определить некую измеряемую среднюю точку, между этими значениями, нельзя. Так же невозможно применить переход от width: auto до width: 1000px. Исключением является свойство visibility, можно осуществить переход от значения visible к hidden.

- transition-duration — определяет время перехода, указывается в s (*секунды*) или в ms (*миллисекунды*). Является обязательным свойством, при котором будет создаваться переход. Если в свойстве transition-property указано несколько свойств, длительность перехода можно указать разную для каждого свойства, через запятую:

  CSS-код:

```
a{
    text-decoration: none;
    display: inline-block;
    font-size:  2rem;
    background-color: #003fa3;
    border-radius: 5px;
    color: #ffffff;
    padding: 15px 25px;
    transition-property: background-color, font-size;
```

```
    transition-duration: 0.8s, 1s;
    transition-timing-function:  ease;
    transition-delay: 0s;
}
```

- transition-timing-function — временная функция, которая определяет, как будут изменяться, указанные свойства. Существует несколько стандартных функций. Также возможно написать свою собственную функцию, используя кривые Безье. На этом сайте https://cubic-bezier.com/ можно сравнить временные функции и увидеть, как они отличаются друг от друга.
  - ▹ cubic-bezier(x1,x2,x3,x4) — значение для собственной функции. Значения x1 и x3 от 0 до 1.

  Стандартные функции:
  - ▹ ease или cubic-bezier(0.25, 0.1, 0.25, 1.0) — переход начинается быстро, к концу скорость увеличивается. Используется по умолчанию;
  - ▹ linear или cubic-bezier(0.0, 0.0, 1.0, 1.0) — скорость линейная, постоянная;
  - ▹ ease-in или cubic-bezier(0.42, 0, 1.0, 1.0) — переход начинается медленно, к концу скорость увеличивается;
  - ▹ ease-out или cubic-bezier(0, 0, 0.58, 1.0) — переход начинается быстро, к концу скорость уменьшается;
  - ▹ ease-in-out или cubic-bezier(0.42, 0, 0.58, 1.0) — переход медленно начинается, к середине времени увеличивается и к концу опять уменьшается. Функция похожа на ease, более выраженное замедление;

Если в свойстве transition-property указано несколько свойств, разные временные функции для каждого свойства можно указывать через запятую:

CSS-код:

```
a{
    text-decoration: none;
    display: inline-block;
    font-size:  2rem;
    background-color: #003fa3;
    border-radius: 5px;
    color: #ffffff;
    padding: 15px 25px;
    transition-property: background-color, font-size;
    transition-duration: 0.8s, 1s;
    transition-timing-function:  ease, linear;
    transition-delay: 0s;
}
```

- transition-delay — определяет задержку до начала перехода, указывается в s или в ms. Если значение не указывается, то по умолчанию оно равно 0 и переход создаётся сразу. Если в свойстве transition-property указано несколько свойств, задержку для каждого свойства можно указывать через запятую:

  CSS-код:

```
a{
    text-decoration: none;
    display: inline-block;
    font-size:  2rem;
    background-color: #003fa3;
    border-radius: 5px;
```

```
    color: #ffffff;
    padding: 15px 25px;
    transition-property: background-color, font-size;
    transition-duration: 0.8s, 1s;
    transition-timing-function:  ease, linear;
    transition-delay: 0s, 0.2s;
}
```

Объявление перехода для нескольких CSS-свойств, с помощью сокращенного свойства transition:

CSS-код:

```
a{
    text-decoration: none;
    display: inline-block;
    font-size:  2rem;
    background-color: #003fa3;
    border-radius: 5px;
    color: #ffffff;
    padding: 15px 25px;
    transition: background-color 0.8s ease 0s,
                font-size 1s linear 0.2s;
}
```

Если значения по умолчанию transition-delay (0s) и transition-timing-function (ease) не планируется изменять, то их можно не указывать в правиле CSS. В этом случае правило будет выглядеть:

CSS-код:

```
a{
    /* все необходимые стили */
    transition: background-color 0.8s,  font-size 1s;
}
```

Если планируется осуществлять переход для всех изменяемых свойств элемента, в одинаковом временном промежутке, то достаточно указать только время перехода:

CSS-код:

```
a{
    /*все необходимые стили*/
    transition: 1s;
}
```

Пример

«Поднимающаяся шторка» при наведении на блок. Это может быть кнопка перехода к какому-либо подразделу сайта.

Рисунок 10. Элементы в начальном состоянии, без наведения

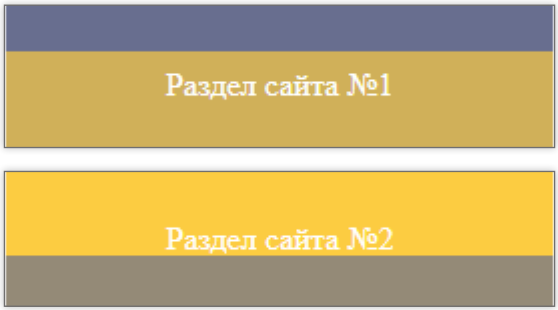

Рисунок 11. Элементы в состоянии — при наведении. При наведении курсора мыши, из нижней границы элемента, плавно появляется полупрозрачный дополнительный фон («шторка»)

HTML-код:

```
<div class="block bl-1">
    <a href="#">Раздел сайта №1</a>
</div>

<div class="block bl-2">
    <a href="#">Раздел сайта №2</a>
</div>
```

CSS-код:

```
/* Добавляем стили для блока, содержащего в себе ссылку */
.block{
    display: inline-block;
    height: 80px;
    width: 20vw;
    position: relative;
}

/* Добавляем стили для ссылки в блоках */
.block a{
    display: block;
    width: 100%;
    height: 100%;
    color: #fff;
    text-align: center;
    text-decoration: none;
    padding: 2em 0;
    position: absolute;
    z-index: 2;
}

/* Цвет фона для каждого блока */
.bl-1{
    background: #686E8F;
}
```

```
.bl-2{
    background: #FCCC41;
}

/*
   С помощью псевдоэлемента ::before добавим «подни-
   мающуюся шторку» для каждого блока со своим цветом
*/
.bl-1::before {
    content: '';
    background-color: rgba(252, 204, 65, 0.7);
    position: absolute;
    bottom: 0;
    left: 0;
    right: 0;
    height: 0%;
    transition: 1s;
}

.bl-2::before {
    content: '';
    background-color: rgba(104, 110, 143, 0.7);
    position: absolute;
    bottom: 0;
    left: 0;
    right: 0;
    height: 0%;
    transition: 1s;
}

/* При наведении «шторка» будет высотой 100% */
.bl-1:hover::before, .bl-2:hover::before{
    height: 100%;
}
```

Свойство transition добавлено для исходного, невыделенного элемента (*код во вложении к уроку, в файле с названием «ButtonSecondBackground.html»*).

CSS-трансформация. Свойство Transform

С помощью трансформаций (свойство transform) элемент можно увеличить в размере, вращать, наклонять, сдвигать, не затрагивая другие элементы на странице.

Часто трансформации используются вместе с переходами. Переходы позволяют сделать процесс трансформации элемента плавным.

Существуют 2D и 3D-типы трансформации. Рассмотрим сначала 2D трансформации. Трансформация происходит в двух осях X (по горизонтали) и Y (по вертикали).

Для преобразования элемента используется свойство transform, которое определяет тип трансформации элемента и свойство transform-origin, которое определяет исходную точку начала трансформации.

Свойство transform-origin

Значения по умолчанию: 50% 50% 0. Первое значение указывает координату x, перемещение элемента по горизонтали. Второе значение указывает координату y, перемещение элемента по вертикали. Эти значения вычисляются от верхнего левого угла элемента. Третье значение задает z-смещение и используется при 3D-преобразованиях. Значения можно указывать в единицах измерения длины элемента, в процентах или ключевыми словами center, left, right, top, bottom. z-смещение задавать не обязательно.

Если преобразование элемента предполагается относительно его центра (transform-origin: 50% 50% — значение по умолчанию), объявлять данное свойство не нужно.

В зависимости от определения исходной точки, с помощью свойства transform-origin, например, вращение (transform: rotate();) будет иметь разный эффект:

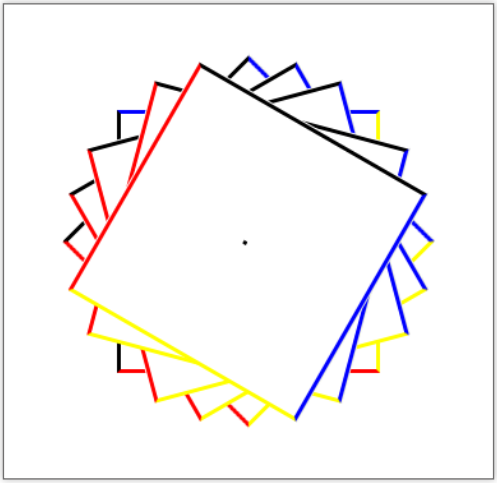

Рисунок 12. Значение по умолчанию: «transform-origin: 50% 50%»

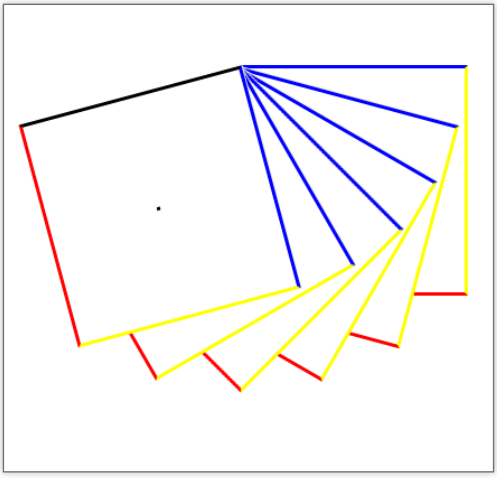

Рисунок 13. transform-origin: 0 0

Функции свойства transform

- translate(x,y) — изменение местоположения элемента на значение x слева направо, на значение y сверху вниз. В случае указания отрицательных значений, сдвиг будет направлен справа налево и снизу вверх:

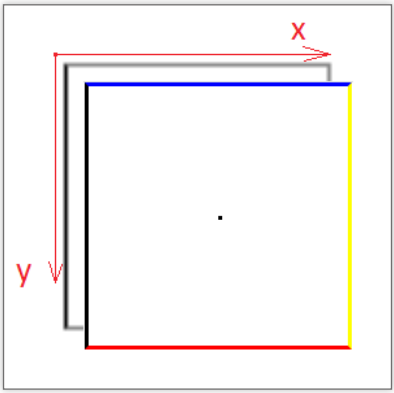

Рисунок 14. Смещение элемента по осям X и Y.

HTML-код:

```
<div class="example">
  <div class="point"></div> <!-- Центральная точка -->
</div>
```

CSS-код:

```
/*Стили для квадрата с разными по цвету границами*/
.example{
    width: 200px;
    height: 200px;
    border: 3px solid black;
    border-top-color: blue;
    border-right-color: yellow;
```

```
    border-bottom-color: red;
    transform: translate(10px, 10px);
}

/*Стили для точки в центре*/
.point{
    width: 3px;
    height: 3px;
    background-color: black;
    position: relative;
    top: 50%;
    left: 50%;

}
```

Свойство transform: translate(10px, 10px); сдвигает элемент по осям x и y на 10px по отношению к своему первоначальному положению. Допустимые значения — единицы измерения размеров элемента или проценты (*пример кода находится во вложении к методпакету, в файле с названием «translate.html»*).

- translateX(x) — похожа на функцию translate(x,y), однако, для неё указывается одно значение для сдвига элемента по оси x, влево или вправо.
- translateY(y) — похожа на функцию translate(x,y), однако, для неё указывается одно значение для сдвига элемента по оси y, вверх или вниз.
- scale(x,y) — масштабирует элемент. Если указаны значения от 0 до 1, то элемент уменьшается. Если значения больше 1, то элемент увеличивается в масштабе. При отрицательных значениях отображает элемент зеркально. Следует отметить, что при увеличении

элемента, качество отображаемого элемента может ухудшиться. Действует для элементов блочного типа (display:block; display:inline-block;). Почему только для таких элементов? Так как масштабирование — это одновременное увеличение ширины и высоты элемента. Изменять ширину/высоту элемента можно для элементов с блочным или строчно-блочным представлением. Ширина и высота строчных элементов рассчитывается в соответствии с содержимым строчного элемента.

Свойство transform: scale(1.2,1.2); увеличит кнопку при наведении в 1,2 раза по высоте и по ширине

Рисунок 15. В начальном состоянии

Рисунок 16. При наведении курсора мыши

HTML-код:

```
<a href="#">Позвонить</a>
```

CSS-код:

```
/*Стили для ссылки-кнопки*/
a{
    display: inline-block;
```

```
    border-radius: 5px;
    padding: 15px 30px;
    background-color: blue;
    color: #fff;
    font-weight: bold;
    text-transform: uppercase;
    text-decoration: none;
}

a:hover{
    transform: scale(1.2,1.2);
}
```

- scaleX(x) — масштабирует элемент по горизонтали. Если указаны значения от 0 до 1, то элемент уменьшается. Если больше 1, то увеличивается в масштабе. При отрицательных значениях отображает элемент зеркально. Следует отметить, что при увеличении элемента, качество может ухудшиться.
- scaleY(y) — масштабирует элемент по вертикали. Если указаны значения от 0 до 1, то элемент уменьшается. Если больше 1, то увеличивается в масштабе. При отрицательных значениях отображает элемент зеркально. Следует отметить, что при увеличении элемента, качество может ухудшиться.
- rotate() — вращает элемент вокруг исходной точки (за расположение исходной точки отвечает свойство transform-origin). Единицы измерения: градусы (deg). При положительном значении — элемент поворачивается по часовой стрелке.

При отрицательном — элемент поворачивается против часовой стрелки. При указании угла поворота более 360 град, элемент будет вращаться на указанное значение, например, при transform: rotate(3600deg), элемент совершить 10 оборотов вокруг исходной точки.

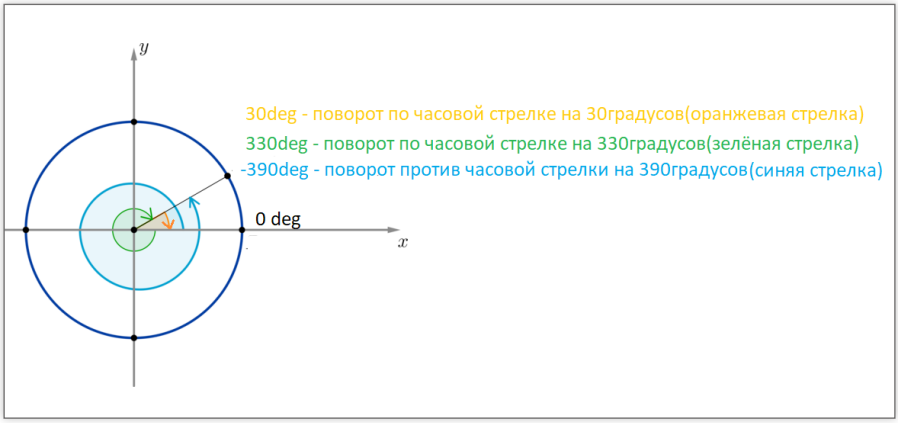

Рисунок 17. Повороты против часовой стрелки/ по часовой стрелке

Создадим элемент страницы «Обратная связь». В начальном состоянии, элемент находится у левой границы окна браузера и видна только розовая часть с текстом «обратная связь» (рис. 18).

При наведении курсора мыши, плавно «выезжает» окно с инструкцией (в примере будет задействован текст «заполнитель» Lorem ipsum) (рис. 19).

На данном этапе, создадим элемент страницы «Обратная связь». Плавный «выезд» всего элемента, при наведении курсора мыши, выполним в разделе «Трансформации и Переходы».

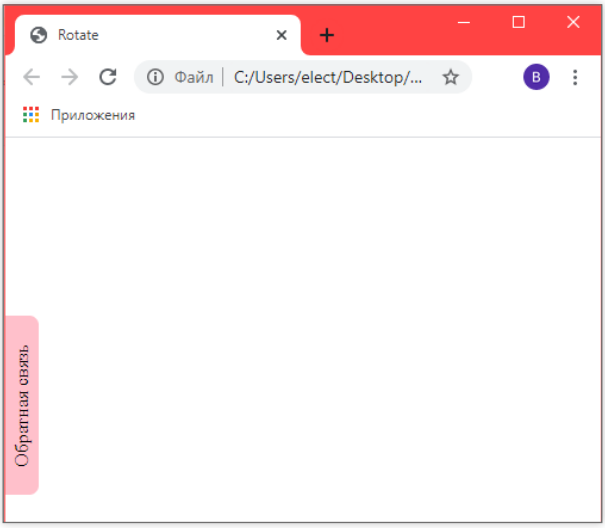

Рисунок 18. Элемент страницы «Обратная связь», при наведении на который появится целое окно

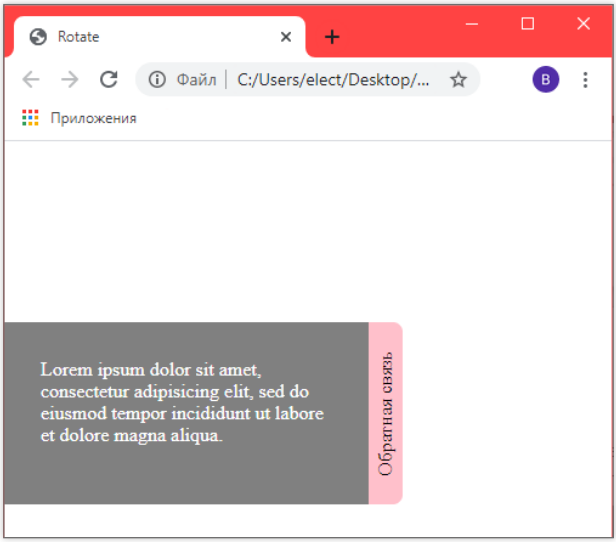

Рисунок 19. Элемент страницы «Обратная связь», появившийся после наведения курсором мыши на текст «Обратная связь»

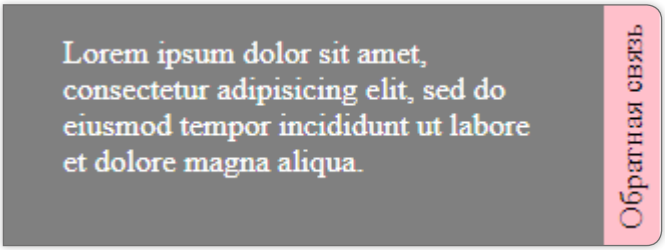

Рисунок 20. Элемент страницы «Обратная связь»

Элемент страницы «Обратная связь», появившийся после наведения курсором мыши на текст «Обратная связь».

Элемент «Обратная связь» (псевдоэлемент ::after) будет выполнен с помощью свойства «transform: rotate(-90deg);» (*полный код примера находится во вложении в метод. пакету, в файле с названием rotate.html).*

Сделаем HTML — разметку для элемента страницы «Обратная связь».

HTML-код:

```
<div id="callback">
    <p>Lorem ipsum dolor sit amet, consectetur
       adipisicing elit, sed do eiusmod tempor
       incididunt ut labore et dolore magna aliqua.
    </p>
</div>
```

Стили для элемента страницы «Обратная связь»:

CSS-код:

```
/* Стили для блока с тестом */
#callback{
    width: 300px;
```

```
    height: 150px;
    background-color: grey;
}
```

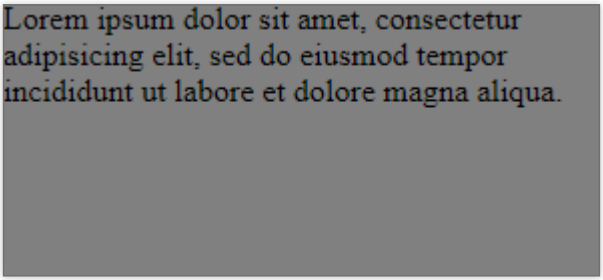

Рисунок 21

```
/* стили для абзаца */
#callback p{
    color: #fff;
    padding: 30px;
}
```

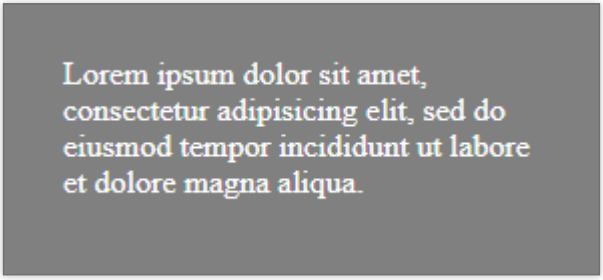

Рисунок 22

Перевернутый текст «Обратная связь» , при наведении на который, будет выезжать окно, выполним с помощью псевдоелемента ::after. Обязательным свойством для

псевдоэлементов является свойство — content. Псевдоэлемент ::after имеет строковое представление, поэтому в значении свойства Content добавим текст ‘Обратная связь’:

```
/* Стили для пседвоэлемента */
#callback:after{
    content: 'Обратная связь';
    background-color: pink;
}
```

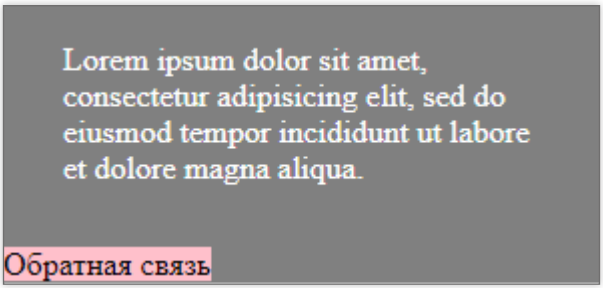

Рисунок 23

Разместим псевдоэлемент ::after, в конце блока с текстом, дополнив стили:

```
/*Стили для пседвоэлемента*/
#callback:after{
    content: 'Обратная связь';
    background-color: pink;
    text-align: center;
    border-radius: 0 0 8px 8px;
    padding: 5px 0;
    width: 150px; /* ширина псевдоэлемента равна
                     высоте блока, тк будет поворот */
    position: absolute; /* позиционирование абсолютное */
```

```
        left: 300px; /* позиция слева равен ширине
                        блока #callback */
    top: 150px; /*позиция сверху равна высоте блока*/
}
```

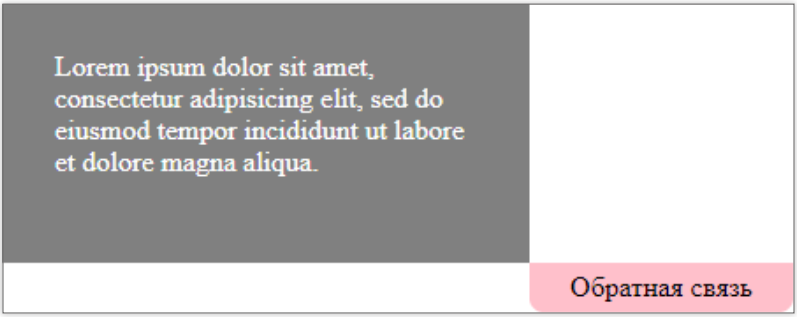

Рисунок 24

Элемент должен совершить вращение против часовой стрелки на 90 градусов, а значит необходимо указать отрицательное значение для функции rotate(-90deg);

```
/* Стили для пседвоэлемента */
#callback:after{
    content: 'Обратная связь';
    background-color: pink;
    text-align: center;
    border-radius: 0 0 8px 8px;
    padding: 5px 0;
    width: 150px; /* ширина псевдоэлемента равна
                     высоте блока, тк будет поворот */
    position: absolute; /* позиционирование абсолютное */
    left: 300px; /* позиция слвеа равен ширине блока
                    #callback */
    top: 150px; /* позиция сверху равна высоте блока */
    transform: rotate(-90deg);
}
```

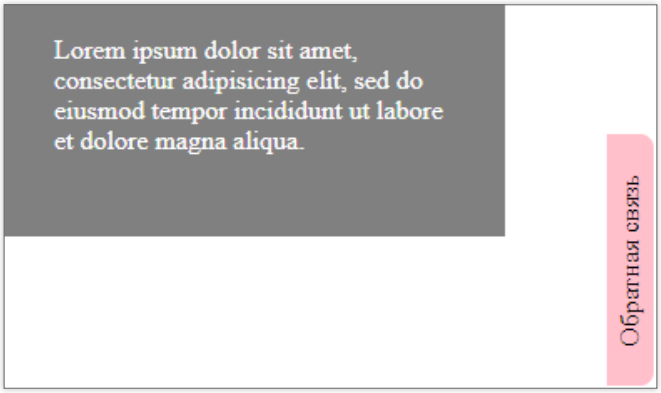

Рисунок 25

Однако, элемент находится не там где надо, потому что сейчас исходная точка определена по центру вращаемого элемента и координаты её расположения равны по умолчанию «transform-origin: 50% 50%;». Вращение происходит относительно этой точки:

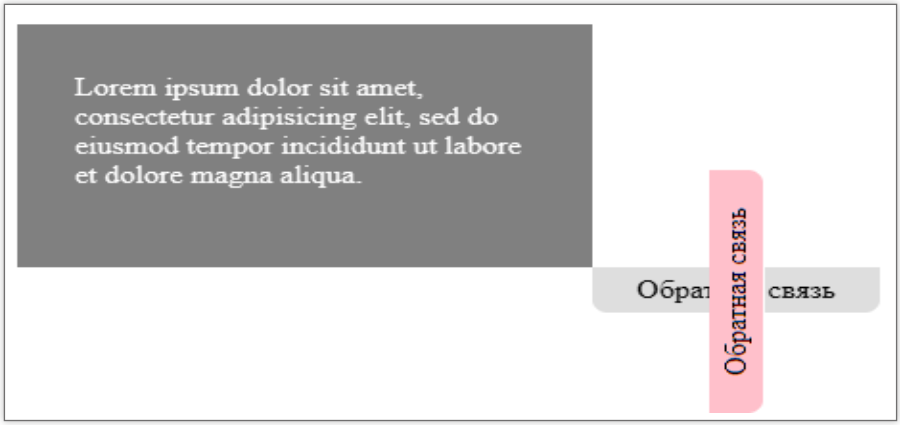

Рисунок 26

Необходимо изменить положение исходной точки, с помощью свойства transform-origin и задать значения,

которые определят расположение исходной точки в левом верхнем углу элемента «Обратная связь»:

```
/* Стили для пседвоэлемента */
#callback:after{
    content: 'Обратная связь';
    background-color: pink;
    text-align: center;
    border-radius: 0 0 8px 8px;
    padding: 5px 0;
    width: 150px; /* ширина псевдоэлемента равна
                     высоте блока, тк будет поворот */
    position: absolute; /* позиционирование абсолютное */
    left: 300px; /* позиция слвеа равен ширине блока
                    #callback */
    top: 150px; /* позиция сверху равна высоте блока */
    transform-origin: 0 0;
    transform: rotate(-90deg);
}
```

Теперь псевдоэлемент «Обратная связь» совершил вращение относительно левого верхнего угла, на 90 градусов против часовой стрелки:

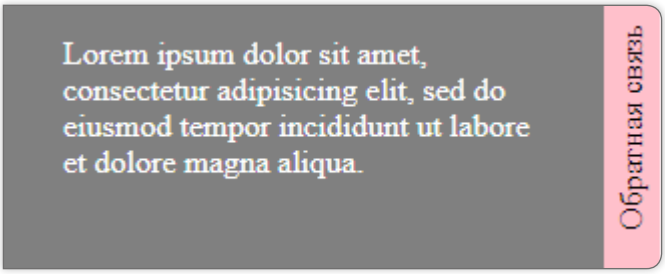

Рисунок 27

- skew(x, y) — определяет наклон элемента вдоль осей X и Y. Если указать только одно значение, то наклон

элемента произойдёт по оси X. Значения указываются в углах, градусах, радианах и оборотах. При положительных значениях наклон совершает по часовой стрелке, при отрицательных против часовой стрелки.

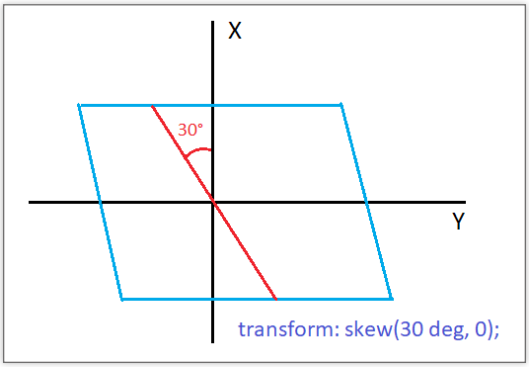

Рисунок 28. Свойство skew, наклон по горизонтали на 30 градусов

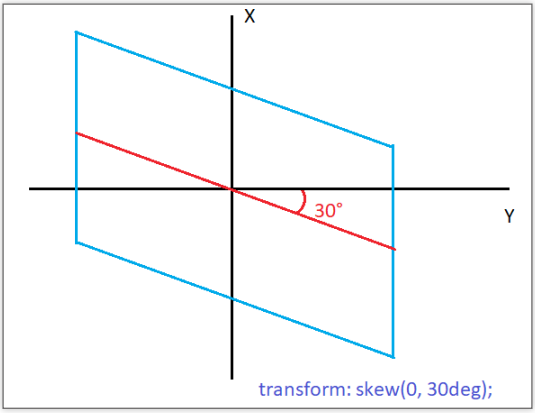

Рисунок 29. Свойство skew, наклон по вертикали на 30 градусов

- skewX(x) — определяет наклон элемента вдоль осей X и Y. Если указать только одно значение, то наклон

элемента произойдёт по оси X. Значения указываются в углах, градусах, радианах и оборотах. При положительных значениях наклон совершает влево, при отрицательных против вправо.

- skewY(y) — определяет наклон элемента вдоль осей X и Y. Если указать только одно значение, то наклон элемента произойдёт по оси X. Значения указываются в углах, градусах, радианах и оборотах. При положительных значениях наклон совершает вверх, при отрицательных вниз.
- matrix — объединяет ряд трансформаций (scale, rotate, skew, traslate) в одно объявление.

  Синтаксис matrix(a, c, b, d, tx, ty), где
  ▹ a — изменение масштаба по горизонтали scaleX;
  ▹ c — наклон по вертикали skewY;
  ▹ b — наклон по горизонтали skewX;
  ▹ d — изменение масштаба по вертикали scaleY;
  ▹ tx — смещение по горизонтали translateX;
  ▹ ty — смещение по вертикали translateY.

Трансформация + переходы

Свойства трансформации часто используется совместно с CSS-переходами. Рассмотрим создание плавно увеличивающейся кнопки (*код примера находится во вложении, файл с названием scale.html).*

Рисунок 30. Кнопка в начальном состоянии

Рисунок 31. Увеличенная кнопка при наведении курсора мыши

Увеличение размера кнопки выполним с помощью свойства «transform: scale(1.2,1.2);» (*масштабирование*). Плавность трансформации выполним с помощью свойства transition.

HTML-код:

```
<a href="#">Позвонить</a>
```

CSS-код:

```
/* Стили для ссылки-кнопки */
a{
    display: inline-block; /* представление inline-
                           block для того, чтобы можно
                           было применить к элементу
                           масштабирование */
    border-radius: 5px;
    padding: 15px 30px;
    background-color: blue;
    color: #fff;
    font-weight: bold;
    text-transform: uppercase;
    text-decoration: none;
}

a:hover{
    transform: scale(1.2,1.2); /* масштабирование
                           элемента по ширине и высоте */
```

```
    transition: 0,5s; /* переход от начального
                         состояния элемента к масшта-
                         бированному при наведении
                         курсора мыши, в течение
                         0,5 секунды */
}
```

Если планируется осуществлять переход со стандартными значениями transition-property, transition-timing-function и transition-delay, то достаточно указать только время перехода — 0,5s.

Указав свойство transition, при наведении курсора мыши (в состоянии :hover), кнопка плавно увеличится. Если курсор мыши убрать, то кнопка моментально вернется в начальное состояние, без плавного перехода. Поэтому необходимо добавить свойство transition и для начального состояния кнопки.

```
a{
    display: inline-block; /* представление inline-
                              block для того, чтобы
                              можно было применить
                              к элементу масшта-
                              бирование */
    border-radius: 5px;
    padding: 15px 30px;
    background-color: blue;
    color: #fff;
    font-weight: bold;
    text-transform: uppercase;
    text-decoration: none;
    transition: 0.5s;
}
```

Рассмотрим ещё один интересный пример сочетания трансформаций и переходов — «поворот» карточки товара при наведении курсора мыши:

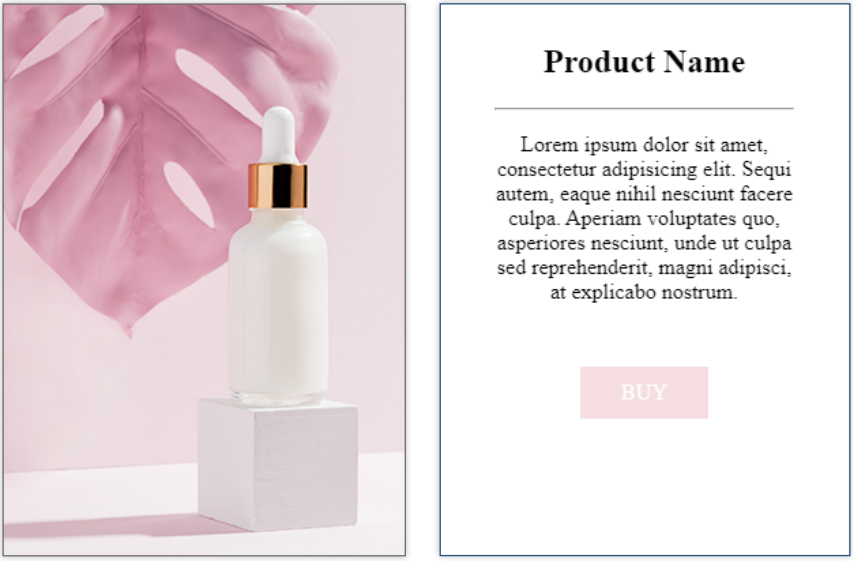

Рисунок 32. Изображение товара в обычном состоянии и обратная сторона карточки, при наведении курсора мыши на изображение

Создадим div с классом card, внутри которого будут находится блоки с классами product (блок с изображением товара) и description (блок с описанием товара и кнопкой «buy») (*полный код примера находится во вложение урока, в файле с названием ProductCard.html*).

HTML-код:

```
<div class="card">
    <div class="product">
        <img src="product.png">
```

```
    </div>
    <div class="description">
        <h2>Product Name</h2>
        <hr>
        <p> Lorem ipsum dolor sit amet, consectetur
            adipisicing elit. Sequi autem, eaque
            nihil nesciunt facere culpa. Aperiam
            voluptates quo, asperiores nesciunt,
            unde ut culpa sed reprehenderit, magni
            adipisci, at explicabo nostrum.</p>
        <a href="#">buy</a>
    </div>
</div>
```

В начальном состоянии карточка товара будет повернута к пользователю стороной с изображением. При наведении курсора мыши на изображение товара, карточка будет поворачиваться по оси X на 180 градусов и перед пользователем появится описание товара с кнопкой «buy».

CSS-код:

```
/*Стили для родительского блока*/
.card {
    width: 300px;
    height: 450px;
    margin: 0 auto;
    position: relative; /* относительное позициониро-
                           вание родительского элемента,
                           чтобы дочерние .product,
                           .description располагать
                           внутри card */
    border: 1px solid #f5dde1;
}
```

Располагаем обе стороны карточки таким образом, чтобы они накладывались друг на друга и занимали родительский элемент на 100%. выровняем содержимое по центру, добавим фон.

```
.product, .description {
    width: 100%; /* ширина 100% родителя */
    height: 100%; /* высота 100% родителя */
    position: absolute; /* абсолютное позиционирование
                           элементов внутри родителя
                           с классом card */
              left: 0;  /* координата позиционирвоания
                           слева 0px*/
              top: 0;   /* координата позиционирвоания
                           сверху 0px */
    text-align:center;
    background-color: #fff;
}

/* Стили для кнопки описательного блока карточки товара */
.description a{
    display:inline-block;
    padding: 10px 30px;
    margin-top:30px;
    text-decoration:none;
    text-transform:uppercase;
    color:#fff;
    background-color: #f5dde1;
}
```

Скроем заднюю часть карточки с помощью свойства «backface-visibility: hidden;».

```
.product, .description {
    backface-visibility: hidden;
}
```

Так как блок с описанием находится в HTML разметке после блока с изображением (рис. 33), то он оказывается поверх блока с изображением:

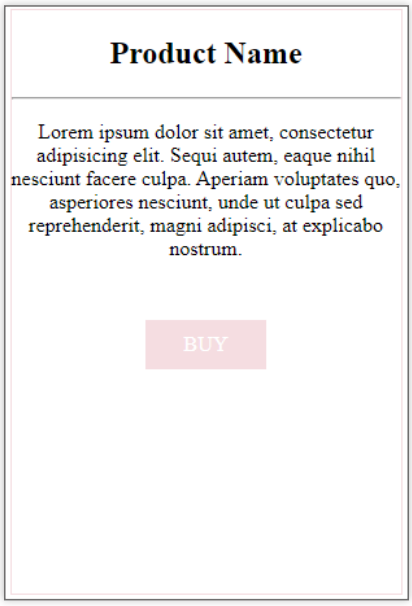

Рисунок 33

Повернём сторону с описанием на 180 градусов по горизонтали и она окажется за лицевой стороной карточки с изображением. В состоянии покоя будет видна только фронтальная сторона карточки — изображение. Так же добавим внутренние поля и свойство «box-sizing: border-box;», чтобы размер блока с описанием высчитывался с учетом внутренних полей и размеров границ:

```
.description {
    box-sizing:border-box;
    padding:40px;
```

```
    transform: rotateY(180deg); /* поворот блока
                с описание на 180 град, прячем его
                в начальном состоянии, без наведения
                курсора мыши */
}
```

Для плавных переходов, при наведении курсора мыши, зададим свойство transition.

```
.product, .description {
    transition: 1s;
}
```

Для того, чтобы увидеть описательную часть товара, необходимо добавить свойства трансформации для блоков с изображением и описанием:

```
.card:hover .product {
    transform: rotateY(180deg); /* разворачиваем блок
                                   с изображением
                                   на 180 градусов */
}

.card:hover .description {
    transform: rotateY(360deg); /* возвращаем блок
                                   с описанием*/
}
```

CSS-анимация. Свойство Animation

Для того, чтобы создать анимацию, её необходимо раскадрировать. Раскадровка используется при создании мультфильмов, раскадрировать можно, например, бег человека или полет птицы (рис. 34, 35).

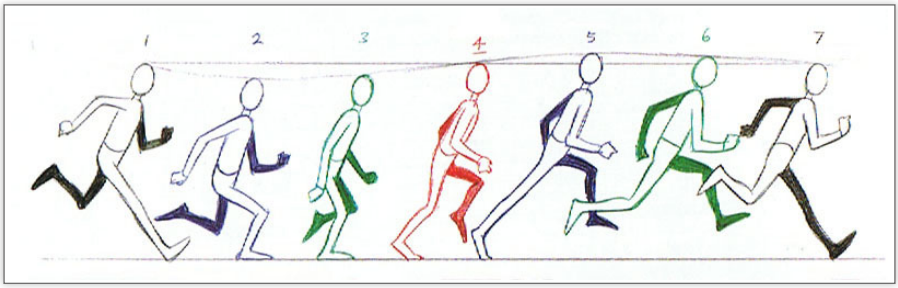

Рисунок 34. Бег человека по кадрам

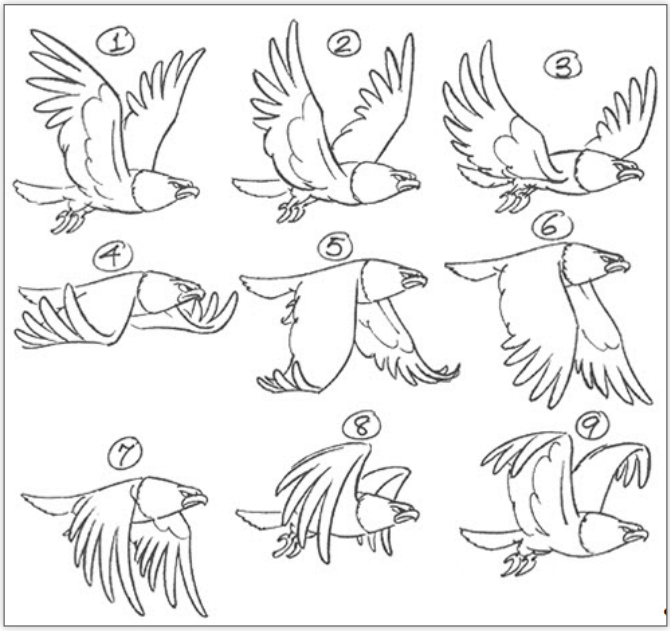

Рисунок 35. Полет рисованного орла по кадрам

Так же происходит и при анимации HTML элементов на странице. Самый простой вид анимации — это два кадра, начальный и конечный. Браузер будет понимать, что необходимо между двумя этими кадрами анимировать элемент, в зависимости от заданных условий.

Кадры создаются с помощью специального правила @keyframes. Сами кадры объявляются ключевыми словами from — «от» или начальный, первый кадр. И to — «до» или конечный кадр, второй и последний кадр. Если количество кадров больше, чем два, то к кадру обращаются, как к отрезкам, выраженным в процентах пройденного пути. Например, три кадра — 0% — начальный первый кадр, 50% — второй кадр, 100% — конечный третий кадр.

Синтаксис:

```
@keyframes name{
    from{
       условие анимации
    }
    to{
       условие анимации
    }
}
```

где, name — имя анимации, создаётся произвольно согласно смыслу анимации.

В качестве первого и простого примера создадим анимацию для заголовка, который будет двигаться слева направо, возвращаться обратно, бесконечное количество раз (*полный код примера находится во вложении, в фале с названием h1Animation.html*).

HTML-код:

```
<h1>Some News</h1>
/*Стили для заголовка*/
h1{
    background-color:#d4c4ff;
```

```
    width: 30vw;
    color: #fff;
}
```

Рисунок 36. Заголовок с фоном

Самый простой способ двигать заголовок — это задать внешний отступ слева margin-left. Т.е. в начальном кадре margin-left будет равен 0 пикселей, в конечном кадре, например, 500 пикселей.

Создадим кадры анимации:

```
@keyframes myAnim{
    from{
        margin-left: 0px;
    }
    to{
        margin-left: 500px;
    }
}
```

Далее необходимо вызвать для элемента, созданную анимацию, с помощью свойства animation:

```
h1{
    animation: myAnim 5s linear alternate infinite;
}
```

где, myAnim — это вызов, созданной с помощью правила @keyframes, анимации, 5s — это скорость анимации от

начального кадра до конечного кадра, linear — это временная функция (линейная), alternate определяет воспроизведение анимации «вперёд-назад», infinite — это количество раз проигрывания анимации, ключевое слово infinite — бесконечное количество раз.

CSS свойство animation — это краткая запись для animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction, animation-fill-mode, animation-play-state.

- animation-name — имя анимации, создается произвольно;
- animation-duration — длительность анимации за один цикл;
- animation-timing-function — временная функция, определяет, как происходит анимация в течение одного цикла. Аналогично свойству transition имеет несколько стандартных функций. Также возможно написать свою собственную функцию, используя кривые Безье. На этом сайте https://cubic-bezier.com/ можно сравнить временные функции и увидеть, как они отличаются друг от друга.
  - ▹ cubic-bezier(x1,x2,x3,x4) — значение для собственной функции. Значения x1 и x3 от 0 до 1.

  Стандартные функции:
  - ▹ ease или cubic-bezier(0.25, 0.1, 0.25, 1.0) — переход начинается быстро, к концу скорость увеличивается. Используется по умолчанию;
  - ▹ linear или cubic-bezier(0.0, 0.0, 1.0, 1.0) — скорость линейная, постоянная;

- ease-in или cubic-bezier(0.42, 0, 1.0, 1.0) — переход начинается медленно, к концу скорость увеличивается;
- ease-out или cubic-bezier(0, 0, 0.58, 1.0) — переход начинается быстро, к концу скорость уменьшается;
- ease-in-out или cubic-bezier(0.42, 0, 0.58, 1.0) — переход медленно начинается, к середине времени увеличивается и к концу опять уменьшается. Функция похожа на ease, более выраженное замедление;

- animation-delay — определяет время задержки перед началом анимации;
- animation-iteration-count — определяет сколько раз будет проигрываться цикл анимации, перед тем как остановиться. Значение infinite определяет бесконечное число циклов.
- animation-direction — определяет направление анимации. Существует несколько стандартных значений:
  - normal — значение по умолчанию, анимация начинается в начальном кадре проходит цикл до конечного кадра и моментально возвращается в начальный кадр;
  - reverse — анимация проигрывается наоборот. Начинается в конченом кадре, проигрывается до начального и сбрасывается в начальном кадре, моментально возвращаясь в конечный кадр;
  - alternate — анимация меняет направление в каждом цикле. От начального кадра идёт вперёд до конченого в одном цикле, в следующем возвращается из конечного кадра в начальный, количество за от числа итераций animation-iteration-count;

- animation-fill-mode — определяет необходимо ли применять стили анимации к элементу до и после окончания анимации. Имеет стандартные значения:
  - ▹ none — стили анимации не будут применены до и после окончания анимации;
  - ▹ forwards — после окончания анимации, элемент сохранит стили анимации конечного ключевого кадра, которые зависит от animation-direction и animation-iteration-count;
  - ▹ backwards — элемент сохранит стили анимации начального кадра, на протяжении всей задержки (animation-delay) до начала анимации, которая определяется значением animation-direction;
  - ▹ both — определяет одновременного применение значений forwards и backwards.
- animation-play-state — определяет состояния паузы и проигрыша анимации. Используется в скриптах, анимацию можно остановить и возобновить с помощью скриптов и проигрываться она начнет с момента остановки, не сначала. Имеет два значения:
  - ▹ runing — анимация проигрывается;
  - ▹ paused — анимация приостановлена.

В сокращенной записи, всех семи свойств, можно опускать значения, которые не планируется задействовать, и они будут установлены в значениях по умолчанию. В нашем примере, использованы свойства animation-name — для вызова созданной анимации для элемента, animation-duration — время проигрывания цикла анимации, animation-timing-function с значением linear (линейная функция, линейная скорость, без скачков),

animation-direction — направление проигрывания анимации, с значением alternate для движения заголовка «вперёд-назад» и animation-iteration-count с значением infinite, для бесконечного проигрывания анимации.

Хороший пример применения свойства animation-iteration-count с значением infinite — бесконечно двигающийся фон с обложками фильмов на сайте https://www.apple.com/tv/.

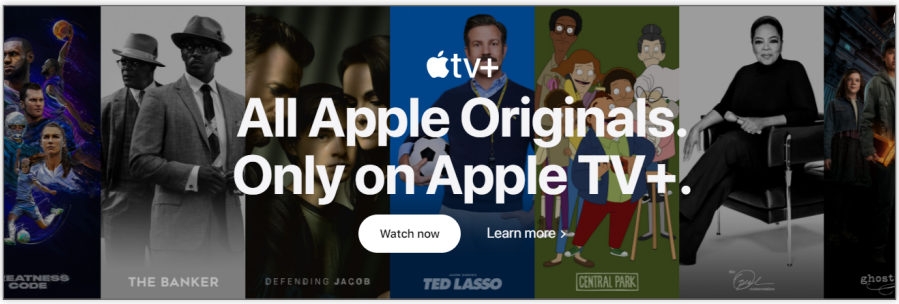

Рисунок 37

Для создания эффекта движущегося фона в ключевых кадрах анимации (@keyframe), описываются свойства background-color и background-image.

Выполним эффект движущихся облаков на голубом фоне (*полный код примера находится во вложении, в фале с названием RuningBackground.html*).

В html-коде ничего не будет, так как фоновое изображение добавляется с помощью css-свойств.

```
/*Стили для фона страницы*/
body{
    background-color: #b6cbe8; /* голубой цвет фона
                                  страницы */
```

```
    background-image:url('cloud.png'); /* изображение
                                  облака в формате png,
                                  с прозрачным фоном */
    }
```

Рисунок 38

«Двигать» фоновое изображение по горизонтали можно с помощью свойства background-position-x на величину равную ширину изображения. Для первого кадра зададим свойство background-position-x: 0px, для конечного кадра background-position-x:500px.

```
@keyframes cloud {
    from{
        background-position-x: 0px;
    }
    to{
        background-position-x: 500px;
    }
}
```

Вызов анимации:

```
body{
    background-color: #b6cbe8;
    background-image: url('cloud.png');
    animation: cloud 10s linear infinite;
}
```

Теперь облака будут бесконечно (значение infinite) двигаться слева направо.

Пример предзагрузчика-индикатора для сайта

Предзагрузчики-индикаторы (Loader) предназначены для информирования посетителя сайта о том, что происходит процесс загрузки страницы. Создадим предзагрузчик-индикатор в виде четырёх квадратов, равноудаленных друг от друга. Они будут загораться белым цветом с течением времени. За счёт этого будет создаваться анимация — эффект перемещающегося квадрата по часовой стрелке:

Рисунок 39. Предзагрузчик-индикатор, 4-е кадра

(*Код примера находится во вложении, в файле с названием loader.html*).

Эффект движущегося квадрата выполним с помощью добавления тени. Теней будет четыре штуки. Добавляться они будут для невидимого квадрата размером 10*10 пикселей. Тени добавим с помощью свойства box-shadow. Данное свойство позволяет добавлять несколько теней и располагать их относительно необходимого элемента по осям x и y. Свойство box-shadow будем описывать при создании анимации для ключевых пяти кадров: 0%, 25%, 50%, 75%, 100%. При первом и последнем кадре

(0% и 100%) значение свойства box-shadow будет одинаковым, для завершения цикла анимации.

HTML-коде:

```
<div class="loader-container">
    <div class="loader"></div>
</div>
```

CSS-код:

```
body{
    background-color: #f3edd7;
    margin: 0;
}
/* стили для родительского контейнера
   предзагрузчика-индикатора */
    .loader-container{
    width: 100vw;
    height: 100vh;
}

/* Стили для предзагрузчика-индикатора, видно его
   не будет на странице */
.loader{
    width: 10px;
    height: 10px;
    position: relative;
    top: 50%;
    margin: auto;
}
```

При объявлении анимации, в ключевых кадрах, добавляем по четыре тени для каждого кадра. Полностью белая тень будет в той позиции box-shadow, которая соответствует порядку ключевого кадра. Значения X и Y для

box-shadow устанавливаем так, чтобы из четырёх теней образовался квадрат, каждая тень в углу.

```
@keyframes loader {
  0%, 100% {
    box-shadow:-10px 15px 0 #ffffff,
               10px 15px 0 rgba(255, 255, 255, 0.2),
               10px 36px 0 rgba(255, 255, 255, 0.2),
              -10px 36px 0 rgba(255, 255, 255, 0.2);
  }

  25% {
    box-shadow:-10px 15px 0 rgba(255, 255, 255, 0.2),
               10px 15px 0 #ffffff,
               10px 36px 0 rgba(255, 255, 255, 0.2),
              -10px 36px 0 rgba(255, 255, 255, 0.2);
  }

  50% {
    box-shadow:-10px 15px 0 rgba(255, 255, 255, 0.2),
               10px 15px 0 rgba(255, 255, 255, 0.2),
               10px 36px 0 #ffffff,
              -10px 36px 0 rgba(255, 255, 255, 0.2);
  }

  75% {
    box-shadow:-10px 15px 0 rgba(255, 255, 255, 0.2),
               10px 15px 0 rgba(255, 255, 255, 0.2),
               10px 36px 0 rgba(255, 255, 255, 0.2),
              -10px 36px 0 #ffffff;
  }
}
```

Вызываем созданную анимация для элемента loader:

```
.loader{
    animation: loader 1s ease infinite;
}
```

Домашнее задание

Задание 1. Реализовать html-страницу с часами

Создайте аналоговые часы, с движущимися стрелками — часовой, минутной и секундной. Анимация начинается с положения стрелок на 12-ти часах. Секундная стрелка должна проходить полный круг за 60 секунд. Минутная стрелка должна передвинуться на одну минуту, после того, как секундная стрелка прошла полный круг, часовая стрелка соответственно должна проходить круг за час.

Пример конечного результата:

Рисунок 40

Задание 2. Реализовать html-страницу со створками

Изначально на странице отображаются два одинаковых блока, за которыми спрятан текст. При наведении курсором на любой из блоков они медленно (за 2 секунды) раскрываются так, чтобы был виден текст. Если курсор убрать, то створки резко возвращаются к первоначальному состоянию.

Пример первоначального состояния:

Рисунок 41

Пример состояния блока при наведении курсором, когда анимация закончилась:

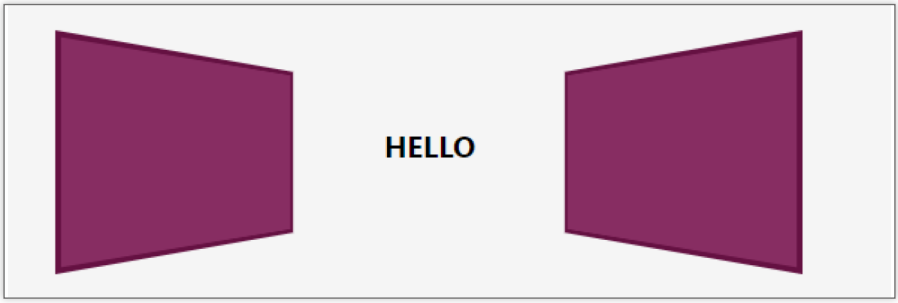

Рисунок 42

Задание 3. Реализовать html-страницу с несколькими блоками текст + картинка

Требования:

- блок состоит из текста и картинки, которые делят доступную ширину поровну;
- блок должен быть фиксированной высоты;
- текст должен находиться посередине своей части;

- картинка должна занимать всю свою часть по ширине и высоте;
- картинка не должна выходить за пределы своей части;
- при наведении мышкой на блок (текст + картинка):
- он должен полностью покрыться полупрозрачной черной плашкой за 0.3 секунды;
- картинка должна плавно (за 1 секунду) увеличиться, но при этом не выйти за пределы своей части.

Изображения необходимо выбрать самостоятельно.

Пример конечного результата (третий блок — это состояние при наведении мышкой):

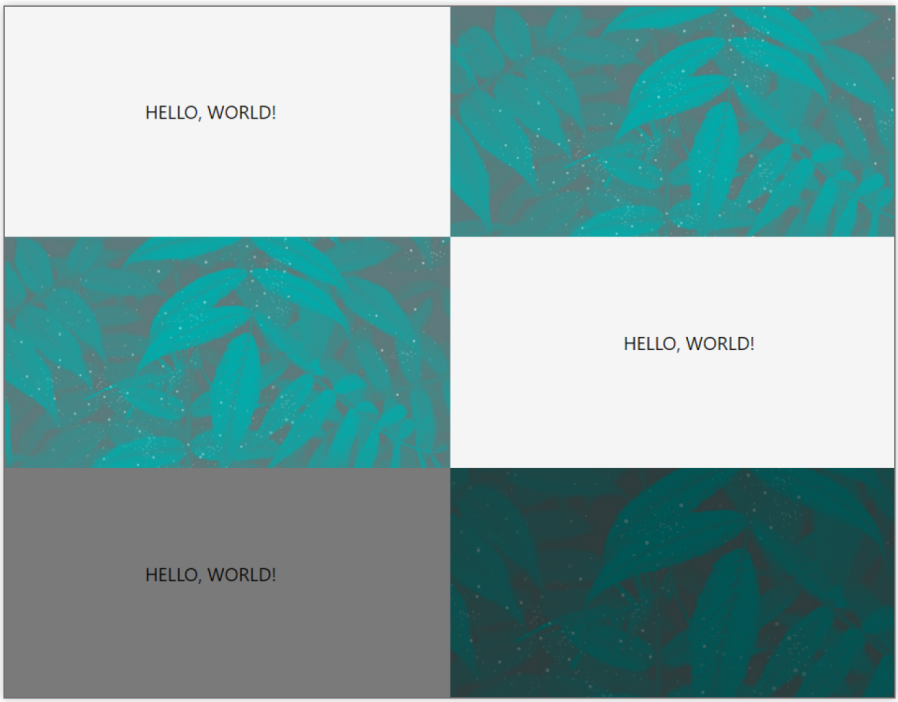

Рисунок 43