**7 Трансформации, переходы и анимации**

**7.1 Трансформации**

Одним из нововведений CSS3 по сравнению с предыдущей версией является встроенная возможность трансформации элемента. Преобразования включают в себя такие действия, как вращение элемента, его масштабирование, наклон или перемещение по вертикали или горизонтали. Чтобы создать преобразования в CSS3, используйте свойство transform.

**7.1.1 Вращение**

Для поворота элемента свойство transform использует функцию rotate:

**transform: rotate(угол\_поворота deg)**

После слова rotate в скобках идет величина угла поворота в градусах. Например, блок поворачивается на 35 градусов.

Далее приведен пример использования вращения.

**<head> <style>**

**div{background-color: #32CD32;**

**width: 100px; height: 100px; margin:6px; padding: 40px 15px;**

**box-sizing: border-box; border: 2px solid black; display: inline-block; }**

**.rotate{ transform: rotate(35deg); }**

**</style> </head>**

**<body style="margin-top: 20px;">**

**<div> Данила Козловский </div>**

**<div class="rotate">Александр Петров </div>**

**<div> Сергей Безруков</div> </body>**

Реализация данного кода представлена на рисунке 7.1.

Можно отметить, что при повороте вращаемый элемент может перекрывать соседние элементы, так как сначала задается положение элементов и только потом происходит поворот.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.1 – Использование поворотов в блоковых элементах

Угол поворота может быть как положительным, так и отрицательным. В случае отрицательного значения вращение производится в противоположную сторону.

**7.1.2 Масштабирование**

Применение масштабирования имеет следующую форму:

**transform: scale(величина\_масштабирования)**

Далее приведен пример использования масштабирования.

<**head>**

**<style>**

**div { background-color: aqua; width: 100px;height: 100px; margin:6px;**

**padding: 45px 15px; box-sizing: border-box;**

**border: 2px solid black; display: inline-block;}**

**.half\_Scale{transform: scale(0.5);}**

**.double\_Scale{transform: scale(2);}**

**</style>**

**</head>**

**<body style="margin-top: 50px;" >**

**<div>2020 год</div><div class="double\_Scale">2021 год</div>**

**<div>2022 год</div> <div class="half\_Scale">2023 год</div>**

**</body>**

Реализация данного кода представлена на рисунке 7.2.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

AI-generated content may be incorrect.

Рисунок 7.2 – Применение масштабирования в CSS

Значение больше 1 приводит к вертикальному и горизонтальному растяжению, а значение меньше 1 - к сжатию. То есть значение 0,5 приводит к уменьшению в два раза, а значение 1,5 - к увеличению в полтора раза.

Также можно установить значения масштаба отдельно для вертикали и горизонтали, как в нижеследующем примере.

**<head> <style>**

**div { background-color: purple;**

**width: 130px;**

**height: 130px;**

**margin:6px;**

**padding: 45px 15px;**

**box-sizing: border-box;**

**border: 2px solid black;**

**display: inline-block; }**

**.scale { transform: scale(2, 0.5); }**

**</style> </head>**

**<body>**

**<div>Москва</div>**

**<div class="scale">Оренбург</div>**

**<div>Санкт-Петербург</div>**

**</body>**

Реализация данного кода представлена на рисунке 7.3.

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, программное обеспечение

AI-generated content may be incorrect.

Рисунок 7.3 – Горизонтальное масштабирование в CSS

В данном случае по горизонтали будет идти масштабирование в 2 раза, а по вертикали - в 0.5 раз. Также можно по отдельности задать параметры масштабирования: функция scaleX() задает изменение по горизонтали, а scaleY() - по вертикали. Например:

**.scale{transform: scaleX(2);}**

Используя отрицательные значения, можно создать эффект зеркального отражения. Далее представлен пример применения такого эффекта.

**<head>  
 <meta charset="utf-8"><title>Использование трансформаций</title>  
 <style>  
 div { background-color: coral;  
 width: 130px;  
 height: 130px;  
 margin:6px;  
 padding: 45px 15px;  
 box-sizing: border-box;  
 border: 2px solid black;  
 display: inline-block; }  
 .scale { transform: scaleX(-1); }  
 </style> </head>  
<body> <div>Телевизор</div>  
 <div class="scale">Холодильник</div>  
 <div>Пылесос</div> </body>**

Реализация данного кода представлена на рисунке 7.4.

Изображение выглядит как текст, снимок экрана, программное обеспечение

AI-generated content may be incorrect.

Рисунок 7.4 - Эффект зеркального отражения

**7.1.3 Перемещение**

Для перемещения элемента используется функция translate:

**transform: translate(offset\_X, offset\_Y)**

Значение offset\_X указывает, на сколько элемент смещается по горизонтали, а offset\_Y - по вертикали. Например, в нижеприведенном примере блок сместился на 30 пикселей вниз и на 50 пикселей вправо.

**<head> <style>  
 img { background-color: silver;  
 width: 130px;  
 height: 130px;  
 margin:6px;  
 box-sizing: border-box;  
 border: 2px solid black;  
 display: inline-block; }  
 .translate { transform: translate(50px, 30px);  
 background-color:blue; } </style> </head> <body>  
<img src="bird1.jpg" alt="">  
<img src="bird2.jpg" alt="" class="translate">  
<img src="bird1.jpg" alt=""> </body>**

Реализация данного кода представлена на рисунке 7.5.

Изображение выглядит как снимок экрана, птица

AI-generated content may be incorrect.

Рисунок 7.5 – Применение перемещения в CSS

В качестве единиц измерения смещения можно применять не только пиксели, но и любые другие единицы измерения длины в CSS - em, % и тд.

С помощью дополнительных функций можно отдельно применять смещения к горизонтали или вертикали: translateX() (перемещение по горизонтали) и translateY() (перемещение по вертикали). Например:

**transform: translateX(30px);**

Кроме положительных значений также можно использовать и отрицательные - они перемещают элемент в противоположную сторону:

**transform: translateY(-2.5em);**

**7.1.4 Наклон**

Для наклона элемента применяется функция skew():

**transform: skew(X, Y);**

Первый параметр указывает, на сколько градусов наклонять элемент по оси X, а второй - значение наклона по оси Y.

Далее приведен пример использования наклонов.

**<head> <style>**

**img{ background-color: silver;  
 width: 130px;**

**height: 130px;  
 margin:6px;  
 box-sizing: border-box;  
 border: 2px solid black;  
 display: inline-block; }  
 .skewe { transform: skew(20deg, 5deg);  
 background-color:aqua;} </style>**

**</head>  
<body> <img src="cat1.jpg" alt="">  
<img src="cat2.webp" alt="" class="skewe">  
<img src="cat3.webp" alt="">**

**</body>**

Реализация данного кода представлена на рисунке 7.6.

Изображение выглядит как снимок экрана, Мелкие и средние кошки, домашняя кошка, млекопитающее

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.6 – Пример использования наклонов в CSS

Для создания наклона только по одной оси для другой оси надо использовать значение 0. Например, наклон на 45 градусов по оси X:

**transform: skew(45deg, 0);**

Или наклон на 45 градусов только по оси Y:

**transform: skew(0,45deg);**

Для создания наклона отдельно по оси X и по оси Y в CSS есть специальные функции: skewX() и skewY() соответственно.

**transform: skewX(45deg);**

Также можно передавать отрицательные значения. Тогда наклон будет осуществляться в противоположную сторону:

**transform: skewX(-30deg);**

**7.1.5 Комбинирование преобразований**

Если надо применить к элементу сразу несколько преобразований, вращение и перемещение, то можно их комбинировать. Например, применение всех четырех преобразований:

**transform: translate(50px, 100px) skew(30deg, 10deg) scale(1.5) rotate(90deg);**

Браузер применяет все эти функции в порядке их следования. То есть в данном случае сначала к элементу применяется перемещение, потом наклон, потом масштабирование и в конце вращение.

**7.1.6 Изменение исходной точки трансформации**

По умолчанию при применении трансформаций браузер использует центр элемента в качестве отправной точки преобразования. Но с помощью свойства transform-origin можно изменить начало. Это свойство принимает значения в пикселях, em и процентах. Также можно использовать ключевые слова, чтобы поставить точку:

* left top: верхний левый угол элемента;
* left bottom: нижний левый угол элемента;
* right top: верхний правый угол элемента;
* right bottom: нижний правый угол элемента.

Далее представлен пример изменения исходной точки трансформации.

**<style> div {background-color: #ccc; width: 100px; height: 100px;**

**margin: 80px 30px; float: left; box-sizing: border-box; border: 1px solid #333;}**

**.transform1 {transform: rotate(-45deg);}**

**.transform2 {transform-origin: left top; transform: rotate(-45deg);}**

**.transform3 {transform-origin: right bottom; transform: rotate(-45deg);} </style></head>**

**<body> <div class="transform1"></div>**

**<div class="transform2"></div>**

**<div class="transform3"></div> </body>**

Реализация данного кода представлена на рисунке 7.7.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.7 – Пример изменения исходной точки трансформации

**7.2 Переходы**

Переход (transition) представляет анимацию от одного стиля к другому в течение определенного периода времени. Чтобы создать переход, сначала необходимы два набора свойств CSS: начальный стиль, который будет иметь элемент в начале перехода, и конечный стиль, который будет результатом перехода.

Далее представлен пример использования переходов.

**<head> <style>**

**div {width: 100px; height: 100px; margin: 40px 30px;**

**border: 1px solid #333; background-color: #ccc;**

**transition-property: background-color; transition-duration: 2s;}**

**div: hover {background-color: red;}**

**</style></head><body> <div></div>**

**</body>**

Реализация данного кода представлена на рисунках 7.8 и 7.9.

Изображение выглядит как текст, снимок экрана, дисплей, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.8 - Переход до

Изображение выглядит как текст, снимок экрана, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.9 - Переход после

В данном примере анимируется свойство background-color элемента div. При наведении курсора мыши на элемент его цвет меняется с серого на красный. А когда указатель мыши отойдет от пространства элемента, вернется первоначальный цвет.

Чтобы указать свойство как анимируемое, его имя передается свойству transition-property.

**transition-property: background-color;**

Далее идет установка времени перехода в секундах с помощью свойства transition-duration:

**transition-duration: 2s;**

Помимо секунд можно устанавливать значения в миллисекундах, например, 500 миллисекунд:

**transition-duration: 500ms;**

Наконец, нужно определить инициатор действия и конечное значение свойства анимированного background-color. Инициатор представляет действие, которое приводит к смене одного стиля на другой. В CSS можно использовать псевдоклассы для запуска перехода. Например, здесь стиль псевдокласса:hover используется для создания перехода. То есть при наведении курсора мыши на элемент div сработает переход.

Помимо псевдокласса :hover можно использовать и другие псевдоклассы, например, :active (ссылка нажимается) или :focus (элемент, получающий фокус).

**7.2.1 Переходы нескольких свойств**

При необходимости можно анимировать сразу несколько свойств CSS. Так, в вышеприведенном примере изменим стили следующем образом:

**div {width: 100px; height: 100px; margin: 40px 30px; border: 1px solid #333;**

**background-color: #ccc;**

**transition-property: background-color, width, height, border-color;**

**transition-duration: 2s;}**

**div:hover {background-color: red; width: 120px; height: 120px; border-color: blue;}**

Здесь анимируются сразу четыре свойства. Причем анимация для них всех длится 2 секунды, но можно для каждого свойства задать свое время:

**transition-property: background-color, width, height, border-color;**

**transition-duration: 2s, 3s, 1s, 2s;**

Подобно тому как в свойстве transition-property через запятую идет перечисление анимируемых свойств, в свойстве transition-duration идет перечисление через запятую временных периодов для анимации этих свойств. Причем сопоставление времени определенному свойству идет по позиции, то есть свойство width будет анимироваться 3 секунды.

Кроме перечисления через запятую всех анимируемых свойств можно просто указать ключевое слово all:

**transition-property: all;**

**transition-duration: 2s;**

Теперь будут анимироваться все необходимые свойства, которые меняют значения в стиле для псевдокласса :hover.

**7.2.2 Функции переходов**

Свойства transition-timing-function позволяют контролировать скорость движения и выполнения анимации. Другими словами, это свойство отвечает за то, как и в какое время анимация будет ускоряться или замедляться.

Это свойство может принимать в качестве значения одну из функций:

* linear: функция линейного сглаживания, свойство изменяется равномерно с течением времени;
* ease: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, обеспечивая более естественное изменение;
* ease-in: функция плавности, при которой вначале происходит только ускорение;
* ease-out: функция плавности, которая ускоряется только в конце анимации;
* ease-in-out: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, обеспечивая более естественные изменения;
* cubic-bezier: для анимации используется кубическая функция Безье.

Далее приведен пример использования функции transition-timing-function со свойством ease-in-out.

**<head> <style>**

**div {width: 100px;height: 100px; margin: 40px 30px; border: 1px solid #333;**

**background-color: #ccc; transition-property: background-color, width;**

**transition-duration: 2s, 10s;transition-timing-function: ease-in-out; }**

**div:hover{ background-color: red; width: 200px;}**

**</style> </head><body>**

**<div></div></body></html>**

Реализация данного кода представлена на рисунках 7.10-7.11.

**Изображение выглядит как текст, снимок экрана, дисплей, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.**

Рисунок 7.10 - Переход до

Изображение выглядит как текст, снимок экрана, логотип, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.11 - Переход после

**7.2.3 Задержка перехода**

Свойство transition-delay позволяет определить задержку перед выполнением перехода:

**transition-delay: 500ms;**

Временной период также указывается в секундах (s) или миллисекундах (ms).

**7.2.4 Свойство transition**

Свойство transition представляет сокращенную запись выше рассмотренных свойств. Например, следующее описание свойств:

**transition-property: background-color;**

**transition-duration: 3s;**

**transition-timing-function: ease-in-out;**

**transition-delay: 500ms;**

Будет аналогично следующей записи:

**transition: background-color 3s ease-in-out 500ms;**

**7.3 Анимация в CSS3**

Анимация дает больше возможности изменить стиль элемента. При переходе есть набор свойств с начальными значениями, которые имеет элемент до начала перехода, и конечными значениями, которые устанавливаются после завершения перехода. Однако при использовании анимации можно иметь не только два набора значений - начальный и конечный, но и множество промежуточных наборов значений.

Анимация предполагает последовательную смену ключевых кадров. Каждый ключевой кадр определяет один набор значений свойств для анимации. И последовательная смена таких кадров фактически будет представлять собой анимацию.

Объявление ключевого кадра в CSS3 имеет следующую форму:

**@keyframes название\_анимации {**

**from {/\* начальные значения свойств CSS \*/ }**

**to {/\* конечные значения свойств CSS \*/}}**

Название анимации отображается после ключевого слова @keyframes. Затем определяются как минимум два ключевых кадра в фигурных скобках. Блок после ключевого слова from объявляет начальный ключевой кадр, а после ключевого слова to блок определяет конечный ключевой кадр. В каждом ключевом кадре определяется одно или несколько свойств CSS, как и при создании обычного стиля.

Далее приведен пример анимации для смены фонового цвета элемента.

**<head> <style>**

**@keyframes backgroundColorAnimation {**

**from {background-color: blue; } to { background-color: purple;} }**

**div{ width: 100px; height: 100px; margin: 40px 30px;**

**border: 1px solid #333; background-color: #ccc;**

**animation-name: backgroundColorAnimation;animation-duration: 2s;}**

**</style> </head> <body> <div>**

**</div> </body>**

В данном случае анимация называется backgroundColorAnimation. Эта анимация обеспечивает переход от синего цвета фона к фиолетовому цвету фона. После завершения анимации будет установлен цвет, определенный для элемента div.

Реализация данного кода представлена на рисунках 8.12-8.13.

Изображение выглядит как текст, снимок экрана, мультимедиа

AI-generated content may be incorrect.

Рисунок 7.12 - Анимация до

Изображение выглядит как снимок экрана, текст, программное обеспечение, мультимедиа

AI-generated content may be incorrect.Изображение выглядит как текст, снимок экрана, мультимедиа

AI-generated content may be incorrect.

Рисунок 7.13 - Анимация после

Чтобы прикрепить анимацию к элементу, используется свойство animation-name в его стиле. Значением этого свойства является имя применяемой анимации.

Также с помощью свойства animation-duration необходимо задать время анимации в секундах или миллисекундах.

При таком определении анимация начнется сразу после загрузки страницы. Однако также можно запускать анимацию на основе действий пользователя. Например, определив стиль псевдокласса :hover, можно указать, что анимация начнется при наведении указателя мыши на элемент.

**@keyframes backgroundColorAnimation {**

**from {background-color: red; }**

**to { background-color: blue; }**

**}**

**div{ width: 100px; height: 100px;margin: 40px 30px;**

**border: 1px solid #333; background-color: #ccc;}**

**div:hover{animation-name: backgroundColorAnimation; animation-duration: 2s;}**

**7.3.1 Множество ключевых кадров**

Как уже выше говорилось выше, анимация кроме двух стандартных ключевых кадров позволяет задействовать множество промежуточных. Для определения промежуточного кадра применяется процентное значение анимации, в котором этот кадр должен использоваться:

**@keyframes backgroundColorAnimation {**

**from {background-color: red; }**

**25%{ background-color: yellow;}**

**50%{ background-color: green;}**

**75%{background-color: blue;}**

**to {background-color: violet;}**

**}**

В данном случае анимация начинается с красного цвета. Через 25% времени анимации цвет меняется на желтый, еще через 25% - на зеленый и так далее (рисунки 7.14-7.16).

Изображение выглядит как текст, снимок экрана, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.14 - До начала анимации

Изображение выглядит как снимок экрана, текст, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст, программное обеспечение, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.15 - Середина анимации

Изображение выглядит как снимок экрана, текст, мультимедиа, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как снимок экрана, текст, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.16 - Конец анимации

Также можно в одном ключевом кадре анимировать сразу несколько свойств.

**@keyframes backgroundColorAnimation {**

**from { background-color: red; opacity: 0.2; }**

**to { background-color: blue; opacity: 0.9; } }**

Также можно определить несколько анимаций, но применять их вместе:

**@keyframes backgroundColorAnimation {**

**from { background-color: red; }**

**to { background-color: blue;  }**

**}**

**@keyframes opacityAnimation {**

**from { opacity: 0.2; }**

**to { opacity: 0.9;  }**

**}**

**div{ width: 100px;**

**height: 100px;**

**margin: 40px 30px;**

**border: 1px solid #333;**

**background-color: #ccc;**

**animation-name: backgroundColorAnimation, opacityAnimation;**

**animation-duration: 2s, 3s; }**

В качестве значения свойства animation-name через запятую перечисляются анимации, и также через запятую у свойства animation-duration задается время этих анимаций. Название анимации и ее время сопоставляются по позиции, то есть анимация opacityAnimation будет длиться 3 секунды.

**7.3.2 Завершение анимации**

В общем случае после завершения временного интервала, указанного у свойства animation-duration, завершается и выполнение анимации. Однако с помощью дополнительных свойств можно переопределить это поведение. Так, свойство animation-iteration-count определяет, сколько раз будет повторяться анимация. Например, три повтора анимации подряд можно задать следующим образом:

**animation-iteration-count: 3;**

Если необходимо, чтобы анимация запускалась бесконечное количество раз, то этому свойству присваивается значение infinite:

**animation-iteration-count: infinite;**

При повторе анимация будет начинаться снова с начального ключевого кадра.

С помощью свойства animation-delay можно определить задержку анимации:

**animation-name: backgroundColorAnimation;**

**animation-duration: 5s;**

**animation-delay: 1s; /\* задержка в 1 секунду \*/**

**7.3.3 Функция плавности анимации**

Как и к переходам, к анимации можно применять теже функции плавности:

* linear: функция линейного сглаживания, изменяется равномерно с течением времени;
* ease: функция сглаживания, при которой анимация ускоряется к середине и замедляется к концу, обеспечивая более естественные изменения;
* ease-in: функция сглаживания, при которой вначале происходит только ускорение;
* ease-out: функция сглаживания, при которой происходит только ускорение в начале;
* ease-in-out: функция сглаживания, при которой анимация ускоряется к середине и замедляется к концу, обеспечивая более естественное изменение.

Для установки функции плавности применяется animation-timing-function:

**@keyframes backgroundColorAnimation {**

**from {background-color: red; }**

**to { background-color: blue; }}**

**div{ width: 100px;height: 100px; margin: 40px 30px; border: 1px solid #333;**

**animation-name: backgroundColorAnimation; animation-duration: 3s;**

**animation-timing-function: ease-in-out;}**

**7.3.4 Свойство animation**

Свойство animation является сокращенным способом определения выше рассмотренных свойств:

**animation: animation-name animation-duration animation-timing-function animation-iteration-count animation-direction animation-delay animation-fill-mode**

Первые два параметра, которые предоставляют название и время анимации, являются обязательными. Остальные значения не обязательны.

Возьмем следующий набор свойств:

**animation-name: backgroundColorAnimation;**

**animation-duration: 5s;**

**animation-timing-function: ease-in-out;**

**animation-iteration-count: 3;**

**animation-direction: alternate;**

**animation-delay: 1s;**

**animation-fill-mode: forwards;**

Этот набор будет эквивалентен следующему определению анимации:

**animation: backgroundColorAnimation 5s ease-in-out 3 alternate 1s forwards;**

**7.3.5 Создание баннера с анимацией**

В качестве примера использования анимации описывается создание баннера.

**<head><style> @keyframes text1**

**{10%{opacity: 1;}**

**40%{opacity: 0;}}**

**@keyframes text2**

**{30%{opacity: 0;}**

**60%{opacity:1;}}**

**@keyframes banner**

**{10%{background-color: #008000;}**

**40%{background-color: #FFFF00;}**

**80%{background-color: #B22222;}}**

**.banner {width: 600px; height: 120px;**

**background-color: #B22222; margin: 0 auto; position: relative;}**

**.text1,.text2 { position: absolute;**

**width: 100%; height: 100%; line-height: 120px;**

**text-align: center;font-size: 40px; color: white; opacity: 0;}**

**.text1 { animation : text1 6s infinite;}**

**.text2 { animation : text2 6s infinite;}**

**.animated { opacity: 0.8; position: absolute; width: 100%;height: 100%;**

**background-color: #B22222; animation: banner 6s infinite;} </style></head><body>**

**<div class="banner"> <div class="animated">**

**<div class="text1">Внимание</div>**

**<div class="text2">Важная информация</div> </div> </div></body>**

Реализация данного кода представлена на рисунках 7.17-7.18.

Изображение выглядит как текст, снимок экрана, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.17 - Баннер до внедрения анимации

Изображение выглядит как текст, снимок экрана, Шрифт, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7.18- Баннер после внедрения анимации

В данном примере одновременно срабатывают три анимации. Анимация “banner” изменяет цвет фона баннера, а анимации text1 и text2 отображают и скрывают текст с помощью настроек прозрачности. Когда первый текст виден, второй не виден и наоборот. Тем самым получается анимация текста в баннере.