**Трансформации, переходы и анимации**

**Трансформации**

Одним из нововведений CSS3 по сравнению с предыдущей версией является встроенная возможность трансформаций элемента.

К трансформациям относятся такие действия, как вращение элемента, его масштабирование, наклон или перемещение по вертикали или горизонтали. Для создания трансформаций в CSS3 применяется свойство **transform**.

### **Вращение**

Для поворота элемента свойство transform использует функцию **rotate**:



После слова  **rotate**  в скобках идет величина угла поворота в градусах. Например, повернем блок на 30 градусов:

|  |  |
| --- | --- |
|  |  |

При этом можно отметить, что при повороте вращаемый элемент может накладываться на соседние элементы, так как сначала происходит установка положения элементов и только затем поворот.

Угол поворота может представлять как положительное, так и отрицательное значение. В случие отрицательного значения поворот производится в противоположную сторону.

### **Масштабирование**

Применение масштабирования имеет следующую форму:



Значение больше 1 приводит к растяжению по вертикали и горизонтали, а меньше 1 - к сжатию. То есть значение 0.5 приводит к уменьшению в два раза, а значение 1.5 - к увеличению в полтора раза.

Используем масштабирование:

|  |  |
| --- | --- |
|  |  |

Можно также задать величины масштабирования отдельно для вертикали и горизонтали:

|  |  |
| --- | --- |
|  | 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:



Используя отрицательные значения, можно создать эффект зеркального отражения:

|  |  |
| --- | --- |
|  |  |

### **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.

К примеру, сместим блок на 30 пикселей вниз и на 50 пикселей вправо:

|  |  |
| --- | --- |
|  |  |

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 можно использовать и отрицательные - они перемещают элемент в противоположную сторону:

transform: translateY(-2.5em);

### **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.

|  |  |
| --- | --- |
|  |  |

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);**

### **Комбинирование преобразований**

Если необходимо применить к элементу сразу несколько преобразований, скажем, вращаение и перемещение, то мы можем 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.

### **Исходная точка трансформации**

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

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

|  |  |
| --- | --- |
|  |  |

## Переходы

Переход (transition) представляет анимацию от одного стиля к другому в течение определенного периода времени.

Для создания перехода необходимы прежде всего два набора свойств CSS: ***начальный стиль***, который будет иметь элемент в начале перехода, и ***конечный стиль*** - результат перехода. Так, рассмотрим простейший переход:

|  |  |
| --- | --- |
|  | Итак, здесь анимируется свойство  **background-color**  элемента **div**.  При наведении указателя мыши на элемент он будет менять цвет с серого на красный. А при уведении указателя мыши с пространства элемента будет возвращаться исходный цвет. |

Чтобы указать свойство как анимируемое, его название передается свойству  **transition-property**

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

Вообще анимировать можно множество разных свойств, такие как ***color, background-color, border-color***.

Полный список свойств CSS, которые поддаются анимации, можно найти по адресу:

[www.w3.org/TR/css3-transitions/#animatable-properties](http://www.w3.org/TR/css3-transitions/#animatable-properties)

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

|  |  |
| --- | --- |
| 1 | transition-duration: 2s; |

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

transition-duration: 500ms;

И в конце необходимо определить инициатор действия и финальное значение анимируемого свойства ***background-color***.

Инициатор представляет действие, которое приводит к изменению одного стиля на другой.

В CSS для запуска перехода можно применять псевдоклассы. Например, здесь для создания перехода используется стиль для псевдокласса ***:hover***. То есть при наведении указателя мыши на элемент **div**, будет срабатывать переход.

Кроме псевдокласса ***:hover*** можно использовать и другие псевдоклассы, например, ***:active*** (ссылка в нажатом состоянии) или ***:focus*** (получение элементом фокуса).

Также для запуска перехода можно использовать код JavaScript.

### **Анимация набора свойств**

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

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



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 можем просто указать ключевое слово **all**:



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

### **Функции анимации**

Свойства **transition-timing-function** позволяет контролировать скорость хода и выполнение анимации. То есть данное свойство отвечет за то, как и в какие моменты времени анимация будет ускоряться или замедляться.

В качестве значения это свойство может принимать одну из функций:

* **linear**: линейная функция плавности, изменение свойства происходит равномерно по времени
* **ease**: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, предоставляя более естественное изменение
* **ease-in**: функция плавности, при которой происходит только ускорение в начале
* **ease-out**: функция плавности, при которой происходит только ускорение в конце анимации
* **ease-in-out**: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, предоставляя более естественное изменение
* **cubic-bezier**: для анимации применяется кубическая функция Безье

Применим функцию **ease-in-out**:



Для использования кубической Безье в функцию **cubic-bezier** необходимо передать ряд значений:



### **Задержка перехода**

Свойство **transition-delay** позволяет определить задержку перед выполнением перехода:

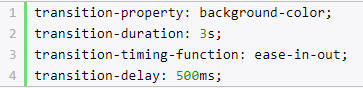
**transition-delay: 500ms;**

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

### **Свойство transition**

Свойство **transition** представляет сокращенную запись выше рассмотренных свойств.

Например, следующее описание свойств:



будет аналогично следующей записи:



## Анимация

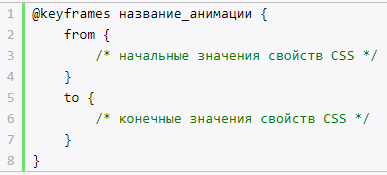
Анимация предоставляет большие возможности за изменением стиля элемента. Предположим, при переходе у нас есть набор свойств с начальными значениями, которые имеет элемент до начала перехода, и конечными значениями, которые устанавливают после завершения перехода. Тогда как при анимации мы можем иметь не только два набора значений - начальные и конечные, но и множество промежуточных наборов значений.

Анимация опирается на последовательню смену ключевых кадров (**keyframes**). Каждый ключевой кадр определяет один набор значений для анимируемых свойств. Последовательная смена таких ключевых кадров фактически будет представлять анимацию.

По сути переходы применяют ту же модель - так же неявно определяются два ключевых кадра - начальный и конечный, а сам переход представляет переход от начального к конечному ключевому кадру.

Единственное отличие анимации в данном случае состоит в том, что для анимации можно определить множество промежуточных ключевых кадров.

В целом объявление ключевого кадра в CSS3 имеет следующую форму:



После ключевого слова **@keyframes** идет имя анимации. Затем в фигурных скобках определяются как минимум два ключевых кадра. Блок после ключевого слова **from** объявляется начальный ключевой кадр, а после клюевого слова **to** в блоке определяется конечный ключевой кадр. Внутри каждого ключевого кадра определяется одно или несколько свойств CSS, подобно тому, как создается обычный стиль.

Например, определим анимацию для фонового цвета элемента:

|  |  |
| --- | --- |
|  | В данном случае анимация называется  **backgroundColorAnimation.**  Анимация может иметь произвольное название.  Эта анимация предоставляет переход от красного цвета фона к синему. Причем после завершения анимации будет устанавливться тот цвет, который определен у элемента div. |

Чтобы прикрепить анимацию к элементу, у него в стиле применяется свойство **animation-name**. Значение этого свойства – название применяемой анимации.

Также с помощью свойства **animation-duration** необходимо задать время анимации в секундах или миллисекундах. В данном случае время анимации - это 2 секунды.

При подобном определении анимация будет запускаться сразу после загрузки страницы. Однако можно также запускать анимацию по действию пользователя.

Например, с помощью определения стиля для псевдокласса **:hover** можно определить запуск анимации при наведении указателя мыши на элемент:



### **Множество ключевых кадров**

Как уже выше говорилось выше, анимация кроме двух стандартных ключевых кадров позволяет задействовать множество промежуточных.

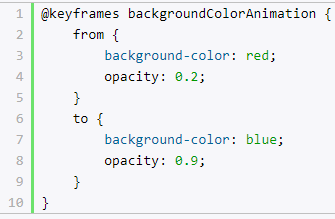
Для определения промежуточного кадра применяется процентное значение анимации, в котором этот кадр должен использоваться:



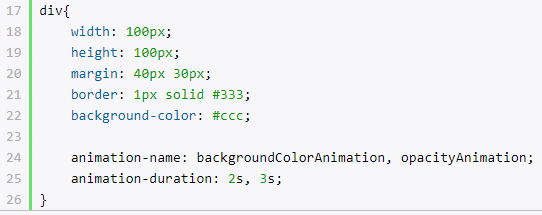
В данном случае анимация начинается с красного цвета.

Через 25% времени анимации цвет меняется на желтый, еще через 25% - на зеленый и так далее.

Также можно в одном ключевом кадре анимировать сразу несколько свойств:



Также можно определить несколько отдельных анимаций, но применять их вместе:



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.

### **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, 3 повтора анимации подряд:



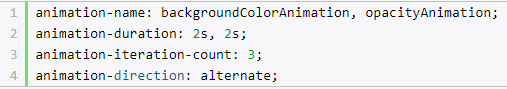
Если необходимо, чтобы анимация запускалась бесконечное количество раз, то этому свойству присваивается значение  **infinite**:



При повторе анимация будет начинаться снова с начального ключевого кадра.

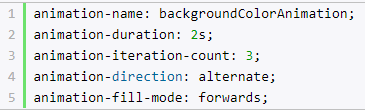
Но с помощью свойства **animation-direction: alternate;** противоположное направление анимации при повторе.

То есть она будет начинаться с конечного ключевого кадра, а затем будет переход к начальному кадру:



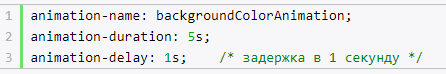
При окончании анимации браузер устанавливает для анимированного элемента стиль, который был бы до применения анимации.

Но свойство  **animation-fill-mode: forwards**  позволяет в качестве окончательного значения анимируемого свойства установить именно то, которое было в последнем кадре:



### **Задержка анимации**

С помощью свойства **animation-delay** можно определить время задержки анимации:



### **Функция плавности анимации**

Как и к переходам, к анимации можно применять все те же функции плавности:

* **linear**: линейная функция плавности, изменение свойства происходит равномерно по времени
* **ease**: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, предоставляя более естественное изменение
* **ease-in**: функция плавности, при которой происходит только ускорение в начале
* **ease-out**: функция плавности, при которой происходит только ускорение в начале
* **ease-in-out**: функция плавности, при которой анимация ускоряется к середине и замедляется к концу, предоставляя более естественное изменение
* **cubic-bezier**: для анимации применяется кубическая функция Безье

Для установки функции плавности применяется свойство **animation-timing-function**:



### **Свойство animation**

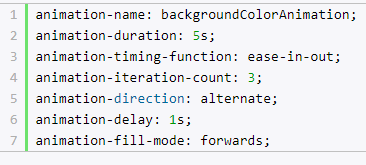
Свойство **animation** является сокращенным способом определения выше рассмотренных свойств:

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:

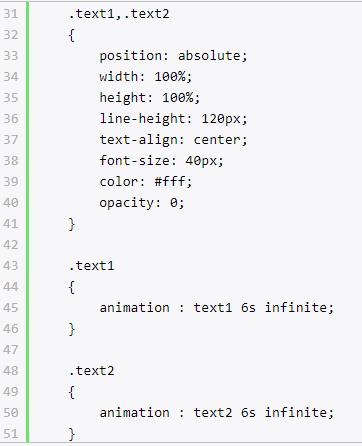


Этот набор будет эквивалентен следующему определению анимации:



### **Создание баннера с анимацией**

В качестве примера с анимацией создадим простейший анимированный баннер:



Здесь 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 мы получаем анимацию текста в баннере.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <title>HTML-баннер</title>      <style type="text/css">      @keyframes text1      {          10%{opacity: 1;}          40%{opacity: 0;}      }      @keyframes text2      {          30%{opacity: 0;}          60%{opacity:1;}      }      @keyframes banner      {          10%{background-color: #008978;}          40%{background-color: #34495e;}          80%{background-color: green;}      }      .banner      {          width: 600px;          height: 120px;          background-color: #777;          margin: 0 auto;          position: relative;      }      .text1,.text2      {          position: absolute;          width: 100%;          height: 100%;          line-height: 120px;          text-align: center;          font-size: 40px;          color: #fff;          opacity: 0;      }        .text1      {          animation : text1 6s infinite;      }        .text2      {          animation : text2 6s infinite;      }        .animated      {          opacity: 0.8;          position: absolute;          width: 100%;          height: 100%;          background-color: #34495e;          animation: banner 6s infinite;      }      </style>  </head>  <body>      <div class="banner">          <div class="animated">              <div class="text1">Только в этом месяце</div>              <div class="text2">Скидки по 20%</div>          </div>      </div>  </body>  </html> |