Урок 6

Эффекты перехода и трансформации CSS3

Создание анимации средствами CSS3. Технология рисования Canvas: попиксельное рисование, текст, тени, градиенты

CSS3-трансформации

transform

Это свойство задает вид преобразования элемента. Оно описывается с помощью функций трансформации, которые смещают элемент относительно его текущего положения на странице или изменяют его первоначальные размеры и форму. Не наследуется.

Допустимые значения:

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

1. matrix(a, b, c, d, x, y).
    a. Смещает элементы и задает способ их трансформации, позволяя объединить несколько функций 2D-трансформации в одной. Допустимы поворот, масштабирование, наклон и изменение положения. Значение a изменяет масштаб по горизонтали.
    b. Значение от 0 до 1 уменьшает элемент, больше 1 – увеличивает.
    c. Значение c деформирует (сдвигает) стороны элемента по оси Y: положительное значение – вверх, отрицательное – вниз.
    d. Значение b деформирует (сдвигает) стороны элемента по оси X: положительное значение – влево, отрицательное – вправо.
    e. Значение d изменяет масштаб по вертикали. Значение меньше 1 уменьшает элемент, больше 1 – увеличивает.
    f. Значение x смещает элемент по оси X: положительное – вправо, отрицательное – влево.
    g. Значение y смещает элемент по оси Y: положительное значение – вниз, отрицательное – вверх.
2. translate(x,y).
    a. Перемещает элемент относительно исходного положения вправо и вниз по указанным координатам x и y, не затрагивая при этом соседние элементы. Отрицательные значения сдвигают элемент влево и вверх.
3. translateX(n)
    a. Сдвигает элемент относительно его обычного положения по оси X.
4. translateY(n)
    a. Сдвигает элемент относительно его обычного положения по оси Y.
5. scale(x,y)
    a. Масштабирует элементы, делая их больше или меньше. Значения от 0 до 1 уменьшают элемент. Первое значение масштабирует элемент по ширине, второе – по высоте. Отрицательные значения отражают элемент зеркально.
    b. scaleX(n) Функция масштабирует элемент по ширине. Если значение больше единицы, элемент становится шире, если значение находится между единицей и нулем – уже. Отрицательные значения отражают элемент зеркально по горизонтали.
    c. scaleY(n) Функция масштабирует элемент по высоте. Если значение больше единицы, элемент становится выше, если значение находится между единицей и нулем – ниже. Отрицательные значения отражают элемент зеркально по вертикали.
6. rotate(угол)

a. Поворачивает элементы на заданное количество градусов. Отрицательные значения от -1deg до -360deg поворачивают элемент против часовой стрелки, положительные – по часовой стрелке. Значение rotate(720deg) поворачивает элемент на два полных оборота.

7. skew(x-угол,y-угол)
   a. Используется для деформирования (искажения) сторон элемента относительно координатных осей. Если указано одно значение, второе будет определено браузером автоматически.
   b. skewX(угол). Деформирует стороны элемента относительно оси X.
   c. skewY(угол). Деформирует стороны элемента относительно оси Y.
   d. initial. Задает значение свойства по умолчанию.
   e. inherit. Наследует значение свойства от родительского элемента.

Практика

Поворот

Трансформации CSS3 – новая, экспериментальная возможность. Поэтому при их использовании нужно указывать несколько версий свойства transform, каждую со своим префиксом разработчика браузеров. Далее приведен пример правила трансформации для вращения элемента со всем его содержимым:

```
<!-- HTML-разметка трансформируемого элемента -->
<figure>
      <img src="/image/7.png" />
      <figcaption>Рисунок</figcaption>
</figure>
```

```
figure {
      -moz-transform: rotate(45deg);
      -webkit-transform: rotate(45deg);
      -o-transform: rotate(45deg);
      -ms-transform: rotate(45deg);
      transform: rotate(45deg);
}
```

Применение трансформации к какому-либо элементу страницы не влияет на другие ее элементы или ее компоновку. Например, если повернуть элемент, то он просто надвинется на смежные с ним элементы.

Обратите внимание, что в предыдущем примере угол поворота задавался в градусах. Спецификация CSS3 предлагает возможность задавать угол и в других математических величинах.

Единицы измерения угла поворота CSS3-трансформации:

Единица измерения	**CSS3-обозначение**	**Описание**	**Пример**
Градусы	*deg*	Угол полной окружности равен 360°	rotate(90deg)

Грады	*grad*	Угол полной окружности равен 400 град (1 град = π/200 радианов)	rotate(100grad)
Радианы	*rad*	Угол полной окружности равен 2π радианов	rotate(1.57rad)
Обороты	*turn*	Угол полной окружности равен одному обороту	rotate(.25turn)

Масштабирование

Для использования анимации масштабирования используется функция scale(). Значение масштаба задается относительно единицы: scale(2) – исходный элемент будет увеличен в два раза, scale(0.5) – элемент уменьшается в два раза. Согласно этим правилам, элемент масштабируется одинаково во все стороны, но вы можете задать направление: X – по горизонтали, Y – по вертикали, Z – глубина масштабирования.

Более того, функцию scale() можно использовать для создания эффекта отражения. Для этого нужно передать ей отрицательное значение. Давайте рассмотрим некоторые примеры масштабирования. Для начала уменьшим нашу картинку с текстом в два раза:

```
figure {
      -moz-transform: scale(0.5);
      -webkit-transform: scale(0.5);
      -o-transform: scale(0.5);
      -ms-transform: scale(0.5);
      transform: scale(0.5);
}
```

Растянем по оси X в два раза:

```
figure {
      -moz-transform: scaleX(2);
      -webkit-transform: scaleX(2);
      -o-transform: scaleX(2);
      -ms-transform: scaleX(2);
      transform: scaleX(2);
}
```

Используем эффект отражения:

```
figure {
      -moz-transform: scaleX(-1);
      -webkit-transform: scaleX(-1);
      -o-transform: scaleX(-1);
      -ms-transform: scaleX(-1);
      transform: scaleX(-1);
}
```

Перемещение

Для перемещения HTML-элемента используется функция translate(x,y), либо ее аналоги для конкретных осей translateX(x) и translateY(y). Эти функции поддерживают отрицательные значения (сдвиг влево или вверх). Ниже показан пример использования трансформации перемещения:

```
figure {
      -moz-transform: translate(50px, -4em);
      -webkit-transform: translate(50px, -4em);
      -o-transform: translate(50px, -4em);
      -ms-transform: translate(50px, -4em);
      transform: translate(50px, -4em);
}
```

Искажение

Трансформация skew() искажает форму элемента. Например, возьмем правильный прямоугольник с закрепленным основанием. Если мы начнем толкать его верхнюю часть в сторону, то она сместится, в то время как основание останется на месте. В результате мы получим параллелограмм.

Значение искажения задается в градусах (можно указывать такие же единицы измерения, как и для трансформации поворота). Ниже показан пример использования трансформации искажения:

```
figure {
      -moz-transform: skewX(50grad);
      -webkit-transform: skewX(50grad);
      -o-transform: skewX(50grad);
      -ms-transform: skewX(50grad);
      transform: skewX(50grad);
}
```

Следующий пример показывает искажение в градусах сразу по двум осям:

```
figure {
      transform:skew(40deg,20deg);
      -webkit-transform:skew(40deg,20deg); // для Chrome и Safari
      -ms-transform:skew(40deg,20deg);     // для IE
}
```

Использование нескольких трансформаций

Вы можете объединить несколько трансформаций в одном правиле, перечислив их через пробел в порядке появления. Ниже приведен пример применения сразу нескольких трансформаций к одному элементу.

```
figure {
      -moz-transform: scale(1.5) translateX(10px) skew(10deg) rotate(0.175rad);
      -webkit-transform: scale(1.5) translateX(10px) skew(10deg)
rotate(0.175rad);
```

```
        -o-transform: scale(1.5) translateX(10px) skew(10deg) rotate(0.175rad);
        -ms-transform: scale(1.5) translateX(10px) skew(10deg) rotate(0.175rad);
        transform: scale(1.5) translateX(10px) skew(10deg) rotate(0.175rad);
}
```

Сначала элемент увеличивается в полтора раза (трансформация scale), потом перемещается на 10 пикселей влево (трансформация translateX) и, наконец, наклоняется и поворачивается (трансформации skew и rotate).

Точка трансформации transform-origin

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

CSS3 transition

Для плавности переходов существует общее свойство transition и отдельные его составляющие (аналогично свойству background, background-color, background-image и т. д.).

Рассмотрим, из чего состоит общее свойство transition:

1. Свойство transition-property. Содержит названия CSS-свойств, к которым будет применен эффект перехода. Не наследуется.
    a. Значения:
        i. **none –** отсутствие свойства для перехода.
        ii. **all –** значение по умолчанию. Применяет эффект перехода ко всем свойствам элемента.
        iii. **свойство –** определяет список css-свойств, участвующих в переходе (перечисляются через запятую).
2. Продолжительность перехода transition-duration Задает промежуток времени, в течение которого должен осуществляться переход. Не наследуется.
    a. Значения:
        i. **время** перехода указывается в секундах или миллисекундах, например, 1s, или 5ms, или 0.3s.
3. Функция перехода transition-timing-function. Свойство задает временную функцию, которая определяет скорость перехода объекта от одного значения к другому.
    a. Значения:
        i. **ease –** функция по умолчанию, переход начинается медленно, разгоняется быстро и замедляется в конце. Соответствует cubic-bezier(0.25,0.1,0.25,1).
        ii. **linear –** переход происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует cubic-bezier(0,0,1,1).
        iii. **ease-in –** переход начинается медленно, а затем плавно ускоряется в конце. Соответствует cubic-bezier(0.42,0,1,1).
        iv. **ease-out –** переход начинается быстро и плавно замедляется в конце. Соответствует cubic-bezier(0,0,0.58,1).
        v. **ease-in-out –** переход медленно начинается и медленно заканчивается. Соответствует cubic-bezier(0.42,0,0.58,1).
        vi. **cubic-bezier(x1, y1, x2, y2) –** позволяет вручную установить значения от 0 до 1 для кривой ускорения.

4. Задержка перехода transition-delay. Необязательное свойство, позволяет сделать так, чтобы изменение свойства происходило не моментально, а с некоторой задержкой. Не наследуется.
    a. **Время** задержки перехода указывается в секундах или миллисекундах.
5. Краткая запись перехода transition. Все свойства, отвечающие за изменение внешнего вида элемента, можно объединить в одно свойство transition.

Практика

Переходы

С помощью псевдоклассов :hover и :focus можно создавать интерактивные эффекты, не прибегая к использованию сценариев JavaScript. Например, чтобы создать меняющуюся кнопку (т. е. кнопку, реагирующую на наведение курсора мыши), достаточно просто предоставить набор новых свойств стиля для псевдокласса :hover. Эти свойства задействуются автоматически, когда пользователь наводит курсор на кнопку.

CSS3 предоставляет более простой способ переходов, позволяющий осуществлять плавный переход от одного набора стилей к другому.

Простой цветовой переход

Чтобы понять принцип работы переходов, рассмотрим пример, в котором изменение цвета кнопки при наведении курсора осуществляется с помощью возможностей перехода CSS3. При наведении курсора ее цвет бы меняется с зеленого на желтый резким прыжком. Но при использовании эффекта перехода зеленый цвет переходит в желтый плавно. Таким же образом происходит и обратный переход при отодвигании курсора.

Смену цвета без эффекта перехода можно реализовать следующим кодом:

```
.slickButton {
    color: white;
    font-weight: bold;
    padding: 10px;
    margin: 20px;
    border: solid 2px gray;
    background: lightgreen;
    cursor: pointer;
}
.slickButton:hover {
    color: black;
    background: yellow;
}
```

Чтобы получить плавное изменение цвета, т. е. переход, нам нужно в только что описанный стиль добавить свойство transition. (Обратите внимание, что это свойство вставляется в обычный стиль – в данном случае в стиль slickButton, – а не в псевдокласс :hover.)

Свойство transition требует установки как минимум двух значений: свойства CSS, которое нужно анимировать, и времени, на протяжении которого нужно выполнить изменение стилей. В данном примере переход применяется к свойству background, а время перехода равно 0.5 секунды:

```
.slickButton {
    /* ... */
    -webkit-transition: background 0.5s, color 0.5s;
    -moz-transition: background 0.5s, color 0.5s;
    -o-transition: background 0.5s, color 0.5s;
}
```

Этот код можно сократить, если нужно установить переход для всех изменяющихся свойств и при одинаковом времени перехода для все хних. В таком случае вместо списка свойств для перехода мы просто используем ключевое слово all:

```
.slickButton {
    /* ... */
    -webkit-transition: all 0.5s;
    -moz-transition: all 0.5s;
    -o-transition: all 0.5s;
}
```

Задержка и комбинирование эффектов перехода

В эффектах перехода можно использовать свойство transition-delay, которое задерживает начало перехода на указанное время:

```
.slickButton {
    /* ... */
    -webkit-transition: all 0.5s;
    -moz-transition: all 0.5s;
    -o-transition: all 0.5s;
    transition: all 0.5s;
    transition-delay: 2s;
}
```

Теперь анимация перехода при наведении указателя мыши будет задерживаться на две секунды. То же самое произойдет и при отведении указателя мыши. Значение свойства transition-delay можно указывать в самом свойстве transition как необязательный параметр:

```
.slickButton {
    /* ... */
    -webkit-transition: all 0.5s 2s;
    -moz-transition: all 0.5s 2s;
    -o-transition: all 0.5s 2s;
    transition: all 0.5s 2s;
}
```

CSS3-анимация

Для создания анимации в CSS3 используется свойство @keyframes.

Данное свойство представляет собой контейнер, в который должны помещаться различные свойства оформления.

Для браузеров Chrome и Safari перед свойством требуется добавить префикс -webkit.

```
@keyframes имяАнимации
{
from {CSS свойства} // Оформление элемента перед началом анимации
to {CSS свойства}   // Оформление элемента после завершения анимации
}
```

После того, как анимация была создана, необходимо добавить к элементу, который вы хотите анимировать, свойство animation и указать в нем имя анимации (1 значение) и время (2 значение), в течение которого она будет выполняться.

Также вы можете устанавливать количество повторов анимации (3 значение):

```
@keyframes anim {
from {margin-left:3px;}
to {margin-left:500px;}
}
#wrap1 {
animation:anim 4s 3;
}
```

Полный листинг анимированной страницы представлен ниже:

```
<html>
<style type='text/css'>
@keyframes anim{
from {margin-left:3px;}
to {margin-left:500px;}
}
@-moz-keyframes anim{
from {margin-left:3px;}
to {margin-left:500px;}
}
@-webkit-keyframes anim{
from {margin-left:3px;}
to {margin-left:500px;}
}
#wrap1{
border:2px #000 solid;
background-color:#7F0055;
height:100px;
width:100px;
font-size:2em;
animation:anim 4s 3;
-webkit-animation:anim 4s 3;
}
</style>
```

```
<body>
<div id="wrap1"></div>
<p><b>Обратите внимание:</b> данная анимация будет повторяться 3 раза.</p>
</body>
</html>
```

Ход выполнения анимации

Вы можете определять ход выполнения анимации не только с помощью ключевых слов from и to (которые использовались в предыдущем примере), но и указывая ключевые точки в %. Например, можно указать, что определенный элемент в начале анимации (0%) должен быть белым, к середине (50%) должен окраситься в оранжевый цвет, а к концу (100%) стать черным.

```
@keyframes anim {
0% {margin-left:3px;margin-top:3px;background-color:#7F0055;}
30% {margin-left:3px;margin-top:250px;background-color:#7F0055;}
60% {margin-left:500px;margin-top:250px;background-color:black;}
100% {margin-left:3px;margin-top:3px;background-color:#7F0055;}
}
#wrap1 {
animation:anim 6s 3;
}
```

Полный листинг анимированной страницы представлен ниже:

```
<html>
<style type='text/css'>
@keyframes anim {
0% {margin-left:3px;margin-top:3px;background-color:#7F0055;}
30% {margin-left:3px;margin-top:250px;background-color:#7F0055;}
60% {margin-left:500px;margin-top:250px;background-color:black;}
100% {margin-left:3px;margin-top:3px;background-color:#7F0055;}
}
@-moz-keyframes anim {
0% {margin-left:3px;margin-top:3px;background-color:#7F0055;}
30% {margin-left:3px;margin-top:250px;background-color:#7F0055;}
60% {margin-left:500px;margin-top:250px;background-color:black;}
100% {margin-left:3px;margin-top:3px;background-color:#7F0055;}
}
@-webkit-keyframes anim {
0% {margin-left:3px;margin-top:3px;background-color:#7F0055;}
30% {margin-left:3px;margin-top:250px;background-color:#7F0055;}
60% {margin-left:500px;margin-top:250px;background-color:black;}
100% {margin-left:3px;margin-top:3px;background-color:#7F0055;}
}
#wrap1 {
border:2px #000 solid;
background-color:#7F0055;
height:100px;
width:100px;
```

```
font-size:2em;
animation:anim 6s 3;
-webkit-animation:anim 6s 3;
}
</style>
<body>
<div id="wrap1"></div>
</body>
</html>
```

CSS3-свойства анимации:

Свойство	Описание
@keyframes	Контейнер для определения анимации
animation	Позволяет задать все значения для настройки выполнения анимации за одно определение
animation-name	Позволяет указать имя анимации
animation-duration	Позволяет задать скорость выполнения анимации в секундах (*по умолчанию имеет значение 0*)
animation-timing-function	Позволяет задать функцию смягчения, отвечающую за плавность выполнения анимации (*по умолчанию имеет значение ease*)
animation-delay	Позволяет задать задержку перед началом выполнения анимации (*по умолчанию имеет значение 0*)
animation-iteration-count	Позволяет задать количество повторов анимации (*по умолчанию имеет значение 1*)
animation-direction	При нечетном значении alternate (*1, 3, 5...*) анимация будет проигрываться в прямом, а при четном (*2, 4, 6...*) – в обратном порядке. По умолчанию это свойство имеет значение normal: при нем анимация всегда проигрывается в прямом порядке.

Длительность анимации animation-duration

Свойство устанавливает длительность анимации. Не наследуется. Значение по умолчанию – 0.

Значения	Описание
Время	Длительность анимации задается в секундах (s) или миллисекундах (ms)
initial	Устанавливает значение свойства по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```
-webkit-animation-duration: 2s;
```

```
animation-duration: 2s;
```

Временная функция animation-timing-function

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

Значения	Описание
ease	Функция по умолчанию, анимация начинается медленно, разгоняется быстро и замедляется в конце. Соответствует cubic-bezier(0.25,0.1,0.25,1)
linear	Анимация происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует cubic-bezier(0,0,1,1)
ease-in	Анимация начинается медленно, а затем плавно ускоряется в конце. Соответствует cubic-bezier(0.42,0,1,1)
ease-out	Анимация начинается быстро и плавно замедляется в конце. Соответствует cubic-bezier(0,0,0.58,1)
ease-in-out	Анимация медленно начинается и медленно заканчивается. Соответствует cubic-bezier(0.42,0,0.58,1)
cubic-bezier(x1, y1, x2, y2)	Позволяет вручную установить значения от 0 до 1. На этом сайте вы сможете построить любую траекторию скорости изменения анимации
step-start	Задает пошаговую анимацию, разбивая анимацию на отрезки, изменения происходят в начале каждого шага. Эквивалентно steps(1, start)
step-end	Пошаговая анимация, изменения происходят в конце каждого шага. Эквивалентно steps(1, end)
steps(количество шагов,start|end)	Ступенчатая временная функция, которая принимает два параметра. Количество шагов задается целым положительным числом. Второй параметр необязательный, указывает момент, в котором начинается анимация. Со значением start анимация начинается в начале каждого шага, со значением end – в конце каждого шага с задержкой. Задержка вычисляется как результат деления времени анимации на количество шагов. Если второй параметр не указан, используется значение по умолчанию end
initial	Устанавливает значение свойства по умолчанию
inherit	Наследует значение свойства от родительского элемента

Шаги анимации

Элемент можно анимировать, используя шаги, т. е. ступенчато. (Примером может послужить секундная стрелка часов, которая сначала двигается, а затем осуществляется задержка на 1 секунду, потом снова двигается, и снова задержка и т. д.) Шаги задаются с помощью функции steps(). Ниже показан пример:

```
transition: all 2000ms steps(3, end);
```

Теперь анимация увеличения кнопки будет происходить рывками. Задержка между рывками в данном случае будет 667 ms (2000/3). Второй параметр функции steps указывает, будет ли рывок выполняться сразу или после задержки.

Анимация с задержкой animation-delay

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

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

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

Синтаксис:

```
-webkit-animation-delay: 2s;
animation-delay: 2s;
```

Повтор анимации animation-iteration-count

Свойство позволяет запустить анимацию несколько раз. Нулевое или отрицательное значение удаляет анимацию из проигрывания. Не наследуется.

Значения	Описание
число	С помощью целого числа задается количество повторов анимации. Значение по умолчанию – 1
infinite	Анимация проигрывается бесконечно
initial	Устанавливает значение свойства по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```
-webkit-animation-iteration-count: 3;
animation-iteration-count: 3;
```

Направление анимации animation-direction

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

Значения	Описание
alternate	Анимация проигрывается с начала до конца, затем в обратном направлении
alternate-reverse	Анимация проигрывается с конца до начала, затем в обратном направлении
normal	Значение по умолчанию, анимация проигрывается в обычном направлении, с начала и до конца
reverse	Анимация проигрывается с конца
initial	Устанавливает значение свойства по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис

```
-webkit-animation-direction: alternate;
animation-direction: alternate;
```

Проигрывание анимации animation-play-state

Свойство управляет проигрыванием и остановкой анимации. Остановить анимацию внутри цикла можно, если использовать это свойство в скрипте JavaScript. Также можно останавливать анимацию в состоянии :hover (при наведении курсора мыши на объект). Не наследуется.

Значения	Описание
paused	Останавливает анимацию
running	Значение по умолчанию, означает проигрывание анимации
initial	Устанавливает значение свойства по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```
-webkit-animation-play-state: paused;
animation-play-state: paused;
```

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

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

Значения	Описание
none	Значение по умолчанию. Состояние элемента не меняется ни до, ни после воспроизведения анимации
forwards	Воспроизводит анимацию до последнего кадра по окончании последнего повтора и не отматывает ее к первоначальному состоянию
backwards	Возвращает состояние элемента после загрузки страницы к первому кадру, даже если установлена задержка animation-delay, и оставляет его там, пока не начнется анимация
both	Позволяет оставить элемент в первом ключевом кадре до начала анимации (игнорируя положительное значение задержки) и задерживать на последнем кадре до конца последней анимации
initial	Устанавливает значение свойства по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```
-webkit-animation-fill-mode: forwards;
animation-fill-mode: forwards;
```

Краткая запись анимации

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

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

Множественные анимации

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

Синтаксис:

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

Технология рисования Canvas

Манипуляции над пикселями

2D Context API предоставляет три метода, которые позволяют выполнять попиксельное рисование: createImageData, getImageData и putImageData.

Пиксели хранятся в объектах типа ImageData. Каждый объект имеет три свойства: width, height и data. Свойство data имеет тип CanvasPixelArray и содержит массив элементов размером width*height*4 байт. Это означает, что каждый пиксель содержит цвет в формате RGBA. Пиксели упорядочены слева направо и сверху вниз, построчно.

Рассмотрим пример отрисовки блока из красных пикселей:

```
// Создадим объект ImageData
var imgd = context.createImageData(50,50);
var pix = imgd.data;
// Пройдемся по всем пикселям и зададим полупрозрачный красный цвет
for (var i = 0; n = pix.length, i < n; i += 4) {
  pix[i] = 255;     // красный канал
  pix[i+3] = 127;   // альфа-канал
}
// Отрисуем объект ImageData в заданных координатах (x,y)
context.putImageData(imgd, 0, 0);
```

Используя возможности метода ImageData, можно проводить различные манипуляции с изображениями. Например, можно создать математическую визуализацию (фракталы, инверсию и т. п.).

Пример создания фильтра для инвертирования цвета изображения:

```
// Получим массив типа CanvasPixelArray по заданным координатам и размерам
var imgd = context.getImageData(x, y, width, height);
var pix = imgd.data;
// Обойдем все пиксели изображения и инвертируем цвет
for (var i = 0, n = pix.length; i < n; i += 4) {
  pix[i] = 255 - pix[i];          // красный канал
  pix[i+1] = 255 - pix[i+1];      // зеленый канал
  pix[i+2] = 255 - pix[i+2];      // синий канал
// i+3 - номер элемента, содержащий альфа-канал
}
// Отрисуем объект ImageData в заданных координатах (x,y)
context.putImageData(imgd, x, y);
```

Текст

Следующие свойства текста доступны для объекта контекста:

- font – определяет шрифт текста (так же, как свойство font-family в CSS);

- textAlign – определяет горизонтальное выравнивание текста. Допустимые значения: start, end, left, right, center. Значение по умолчанию: start;
- textBaseline – определяет вертикальное выравнивание текста. Допустимые значения: top, hanging, middle, alphabetic, ideographic, bottom. Значение по умолчанию: alphabetic.

Есть два метода для вывода текста: fillText и strokeText. Первый отрисовывает текст, заполняя его заливкой стиля fillStyle, другой рисует обводку текста, используя стиль strokeStyle. Оба метода принимают три аргумента: собственно текст и координаты (x,y), в которых его необходимо вывести. Также существует четвертый необязательный аргумент – максимальная ширина блока текста.

Свойства выравнивания текста влияют на позиционирование текста относительно координат его вывода (x,y).

Пример вывода текста:

```
<canvas id="myCanvas6" width="300" height="250">
  <p> </p>
  <p>Альтернативное содержимое, которое будет показано, если браузер не
поддерживает Canvas. В данном случае можно написать сообщение «Ваш браузер не
поддерживает Canvas».</p>
</canvas>
<script>
    var canvas = document.getElementById("myCanvas6");
    var context = canvas.getContext("2d");
context.fillStyle = '#00f';
context.font = 'italic 30px sans-serif';
context.textBaseline = 'top';
context.fillText ('Пример вывода текста с заливкой', 0, 0);
context.font = 'bold 30px sans-serif';
context.strokeText('Пример вывода текста без заливки', 0, 50);
</script>
```

Тени

Shadow API предоставляет четыре свойства:

- shadowColor – определяет цвет тени. Значения допустимы в том же формате, что и в CSS;
- shadowBlur – определяет степень размытия тени в пикселях. Эффект очень похож на гауссово размытие в Photoshop;
- shadowOffsetX и shadowOffsetY – определяет сдвиг тени в пикселях (x, y).

```
<canvas id="myCanvas7" width="600" height="150">
  <p>Альтернативное содержимое, которое будет показано, если браузер не
поддерживает Canvas. В данном случае можно написать сообщение «Ваш браузер не
поддерживает Canvas».</p>
</canvas>
<script>
    var canvas = document.getElementById("myCanvas7");
    var context7 = canvas.getContext("2d");
      context7.shadowOffsetX = 5;
      context7.shadowOffsetY = 5;
      context7.shadowBlur = 4;
```

```
      context7.shadowColor = 'rgba(255, 0, 0, 0.5)';
      context7.fillStyle = '#652';
      context7.fillRect(20, 20, 150, 100);
</script>
```

Градиенты

Свойства fillStyle и strokeStyle также позволяют указывать объекты CanvasGradient вместо обычных цветов CSS, то есть для линий и заливок можно использовать градиенты.

Для создания объектов CanvasGradient используются два метода: createLinearGradient и createRadialGradient. Первый метод создает линейный градиент, а второй – радиальный.

Как только создан объект градиента, можно добавлять в него цвета с помощью метода addColorStop.

```
<canvas id="myCanvas8" width="600" height="220">
  <p>Альтернативное содержимое, которое будет показано, если браузер не
поддерживает Canvas. В данном случае можно написать сообщение «Ваш браузер не
поддерживает Canvas».</p>
</canvas>
<script>
    var canvas = document.getElementById("myCanvas8");
    var context8 = canvas.getContext("2d");
// Нужно указать начальные и конечные координаты (x,y) градиента
var gradient1 = context8.createLinearGradient(0, 0, 100, 100);
// Теперь можно добавлять цвета в градиент
// Первый градиент определяет позицию для цвета в градиенте
// Допустимы значения от 0 (начало градиента) до 1 (конец градиента)
gradient1.addColorStop(0, '#f00');    // красный
gradient1.addColorStop(0.5, '#ff0');  // желтый
gradient1.addColorStop(1, '#00f');    // синий
context8.fillStyle = gradient1;
context8.fillRect(20, 20, 150, 100);
</script>
```

Практическое задание

1. Применить к нескольким элементам на странице эффекты:
   – трансформации (увеличения изображения при наведении);
   – поворота (добавить поворот элемента на 180 или на 360 градусов, при наведении на него);
   – * анимации (по желанию);
   – сделать трансформацию элементов плавной (transition);
   – * применить к произвольному элементу технологию рисования Canvas (по желанию).
2. * Для некоторых элементов применить функцию, которая позволяет изменять скорость перехода в процессе его осуществления.
3. * Использовать для некоторых элементов множественную анимацию.

4. * Усовершенствовать созданные на предыдущих уроках страницы сайта. Изменить шрифты используемые на страницах. При необходимости разместить текстовую информацию, используя колонки.
5. * Усовершенствовать страницы сайта так, чтобы было комфортно их просматривать на мобильных устройствах.

Дополнительные материалы

1. https://webref.ru/css/transition-timing-function.
2. https://developer.mozilla.org/en-US/docs/Web/CSS/transition-property.

Используемая литература

1. http://www.wisdomweb.ru/.
2. https://html5book.ru/.
3. Гоше Х. HTML5. Для профессионалов. СПб.: Питер, 2013. – 496 с.
4. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения. СПб.: Питер, 2012.
5. Макфарланд Д. Большая книга CSS3. СПБ.: Питер, 2014.