Использование CSS-анимации

CSS-анимации позволяют анимировать переходы от одной конфигурации CSS стилей к другой. CSS-анимации состоят из двух компонентов: стилевое описание анимации и набор ключевых кадров, определяющих начальное, конечное и, возможно, промежуточное состояние анимируемых стилей.

Есть три преимущества CSS-анимации перед традиционными способами:

1. Простота использования для простых анимаций; вы можете создать анимацию, не зная JavaScript.
2. Анимации будут хорошо работать даже при умеренных нагрузках системы. Простые анимации на JavaScript, если они плохо написаны, часто выполняются плохо. Движок может использовать frame-skipping и другие техники, чтобы сохранить производительность на таком высоком уровне .
3. Позволяет браузеру контролировать последовательность анимации, тем самым оптимизируя производительность и эффективность браузера. Например, уменьшая частоту обновления кадров анимации в непросматриваемых в данный момент вкладках.

Конфигурирование анимации

Чтобы создать CSS-анимацию вы должны добавить в стиль элемента, который хотите анимировать, свойство `animation` или его подсвойства. Это позволит вам настроить ускорение и продолжительность анимации, а также другие детали того, как анимация должна протекать. Это не поможет вам настроить внешний вид анимации, который настраивается с помощью `@keyframes` (en-US), рассматриваемой далее в Определение последовательности анимации с помощью ключевых кадров.

Свойство `animation` имеет следующие подсвойства:

`animation-name`
: Определяет имя `@keyframes` (en-US), настраивающего кадры анимации.

`animation-duration`
: Определяет время, в течение которого должен пройти один цикл анимации.

`animation-timing-function`
: Настраивает ускорение анимации.

`animation-delay`
: Настраивает задержку между временем загрузки элемента и временем начала анимации.

`animation-iteration-count`
: Определяет количество повторений анимации; вы можете использовать значение `infinite` для бесконечного повторения анимации.

`animation-direction`
: Даёт возможность при каждом повторе анимации идти по альтернативному пути, либо сбросить все значения и повторить анимацию.

`animation-fill-mode`
: Настраивает значения, используемые анимацией, до и после исполнения.

`animation-play-state`
: Позволяет приостановить и возобновить анимацию.

При указании стилей CSS внутри `@keyframes` правило, анимация будет постепенно меняться от текущего стиля к новому стилю в определенное время.

Чтобы получить анимацию для работы, необходимо привязать анимацию к элементу.

В следующем примере анимация "example" привязывается к элементу . Анимация будет длиться 4 секунды, и она будет постепенно менять цвет фона элемента от "Red" на "желтый":

Пример

```
/* The animation code */
@keyframes example {
    from {background-color: red;}
    to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
    width: 100px;
    height: 100px;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
}
```

Примечание: Свойство `animation-duration` определяет, сколько времени должно занять анимация для завершения. Если свойство `animation-duration` не задано, анимация не будет выполняться, так как значение по умолчанию равно 0 секундам.

В приведенном выше примере мы указали, когда стиль изменится с помощью ключевых слов "from" и "to" (который представляет 0% (Start) и 100% (полный)).

Также можно использовать процент. С помощью процента можно добавить любое количество изменений стиля.

В следующем примере изменяется цвет фона элемента < div > при завершении анимации на 25%, завершении 50% и повторном завершении анимации на 100%:

Пример

```
/* The animation code */
@keyframes example {
    0%   {background-color: red;}
    25%  {background-color: yellow;}
    50%  {background-color: blue;}
    100% {background-color: green;}
}

/* The element to apply the animation to */
div {
    width: 100px;
    height: 100px;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
}
```

В следующем примере изменяется как цвет фона, так и положение элемента при завершении анимации на 25%, завершении 50% и снова при завершении анимации 100%:

Пример

```
/* The animation code */
@keyframes example {
    0%   {background-color:red; left:0px; top:0px;}
    25%  {background-color:yellow; left:200px; top:0px;}
    50%  {background-color:blue; left:200px; top:200px;}
    75%  {background-color:green; left:0px; top:200px;}
    100% {background-color:red; left:0px; top:0px;}
}

/* The element to apply the animation to */
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
}
```

Задержка анимации

Свойство `animation-delay` указывает задержку начала анимации.

Следующий пример имеет задержку в 2 секунды перед началом анимации:

Пример

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-delay: 2s;
}
```

Отрицательные значения также разрешены. При использовании отрицательных значений анимация запускается, как если бы она уже воспроизводится в течение *N* секунд.

В следующем примере анимация начнется, как если бы она уже играла в течение 2 секунд:

Пример

```
div {
    width: 100px;
```

```
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-delay: -2s;
}
```

Установить, сколько раз анимация должна выполняться

Свойство `animation-iteration-count` указывает, сколько раз должна выполняться анимация.

В следующем примере анимация будет выполняться 3 раза, прежде чем она остановится:

Пример

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: 3;
}
```

В следующем примере используется значение "Infinite" для того, чтобы анимация продолжалась навсегда:

Пример

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: infinite;
}
```

Запуск анимации в обратном направлении или альтернативные циклы

Свойство `animation-direction` указывает, следует ли воспроизвести анимацию вперед, назад или в альтернативных циклах.

Свойство "направление анимации" может иметь следующие значения:

- `normal` - Анимация воспроизводится как обычная (вперед). Это значение по умолчанию
- `reverse` - Анимация воспроизводится в обратном направлении (назад)
- `alternate` - Анимация сначала разыгрывается вперед, затем назад
- `alternate-reverse` - Анимация сначала воспроизводится назад, а затем пересылается

В следующем примере анимация будет запущена в обратном направлении (назад):

Пример

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-direction: reverse;
}
```

В следующем примере используется значение "альтернативный", чтобы сначала запустить анимацию вперед, а затем назад:

Пример

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: 2;
    animation-direction: alternate;
}
```

В следующем примере используется значение "альтернативный-обратный" для того, чтобы анимация сначала пробежала назад, а затем пересылает:

Пример

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
```

```
    animation-iteration-count: 2;
    animation-direction: alternate-reverse;
}
```

Укажите кривую скорости анимации

Свойство `animation-timing-function` определяет кривую скорости анимации.

Свойство "анимация-время-функция" может иметь следующие значения:

- `ease` - Указывает анимацию с медленным запуском, а затем быстро, а затем закончить медленно (это по умолчанию)
- `linear` - Задает анимацию с одинаковой скоростью от начала до конца
- `ease-in` - Задает анимацию с медленным запуском
- `ease-out` - Задает анимацию с медленным концом
- `ease-in-out` - Задает анимацию с медленным началом и концом
- `cubic-bezier(n,n,n,n)` - Позволяет определить собственные значения в функции кубической Безье

В следующем примере показаны некоторые из различных кривых скорости, которые могут быть использованы:

Пример

```
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
```

Задание режима заливки для анимации

Анимация CSS не влияет на элемент до воспроизведения первого ключевого кадра или после воспроизведения последнего ключевого кадра. Свойство "анимация-режим заполнения" может переопределить это поведение.

Свойство `animation-fill-mode` задает стиль для целевого элемента, если анимация не воспроизводится (до начала, после завершения или и того и другого).

Свойство "анимация-режим заполнения" может иметь следующие значения:

- `none` - Значение по умолчанию. Анимация не будет применять стили к элементу до или после выполнения
- `forwards` - Элемент сохранит значения стиля, заданные последним ключевым кадром (зависит от анимации-направления и анимации-количество итераций)
- `backwards` - Элемент получит значения стиля, заданные первым ключевым кадром (в зависимости от направления анимации), и сохранит это во время анимации-период задержки

- `both` - Анимация будет следовать правилам как вперед, так и назад, расширяя свойства анимации в обоих направлениях

Следующий пример позволяет элементу сохранять значения стиля из последнего ключевого кадра при завершении анимации:

Пример

```
div {
    width: 100px;
    height: 100px;
    background: red;
    position: relative;
    animation-name: example;
    animation-duration: 3s;
    animation-fill-mode: forwards;
}
```

Следующий пример позволяет элементу получить значения стиля, заданные первым ключевым кадром до начала анимации (во время периода задержки анимации):

Пример

```
div {
    width: 100px;
    height: 100px;
    background: red;
    position: relative;
    animation-name: example;
    animation-duration: 3s;
    animation-delay: 2s;
    animation-fill-mode: backwards;
}
```

Следующий пример позволяет элементу получить значения стиля, заданные первым ключевым кадром до начала анимации, и сохранить значения стилей из последнего ключевого кадра при завершении анимации:

Пример

```
div {
    width: 100px;
    height: 100px;
    background: red;
    position: relative;
    animation-name: example;
    animation-duration: 3s;
    animation-delay: 2s;
    animation-fill-mode: both;
}
```

Анимация Сокращенное свойство

В приведенном ниже примере используются шесть свойств анимации:

Пример

```
div {
    animation-name: example;
    animation-duration: 5s;
    animation-timing-function: linear;
    animation-delay: 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}
```

Такой же эффект анимации, как и выше, можно достичь с помощью сокращенного `animation` Свойства:

Пример

```
div {
    animation: example 5s linear 2s infinite alternate;
}
```

Свойства анимации CSS

В следующей таблице перечислены правила @keyframes и все свойства анимации CSS:

Свойство	Описание
@keyframes	Указывает код анимации
animation	Сокращенное свойство для задания всех свойств анимации
animation-delay	Указывает задержку начала анимации
animation-direction	Указывает, следует ли воспроизвести анимацию вперед, назад или в альтернативных циклах
animation-duration	Указывает, сколько времени должно занять анимация для завершения одного цикла
animation-fill-mode	Задает стиль элемента, если анимация не воспроизводится (до начала, после завершения или и то, и другое)
animation-iteration-count	Указывает, сколько раз должна воспроизводиться анимация

animation-name	Указывает имя анимации @keyframes
animation-play-state	Указывает, запущена ли анимация или приостановлена
animation-timing-function	Задает кривую скорости анимации

!!!!!ПОДПИСАТЬ НА КАЖДОЙ СТРАНИЦЕ (ФИ учащегося, группу, краткое описание задания)

Задание 1. Два объекта разных цветов движутся по области 200 на 500 пикселей в противоположном направлении, при встрече идут обратно в начальную точку. время движения 6 сек. Количество повторов минимально 3раза

Задание 2. Два объекта разных меняющихся цветов движутся по области 700 на 700 пикселей в противоположном направлении, проходя через каждый угол, при встрече меняются. время движения 12 сек. Количество повторов минимально 3раза

Задание 3. Имитация свободного падения мяча. Время 8-16сек, минимальное количество отскоков 2.