# «Fullstack» разработка

Лекция 5

Анимация

# Анимация

- Анимация как мини-фильм, где вы в качестве режиссёра даёте инструкции (стилевые правила) вашим актёрам (элементам HTML) для разных сцен (ключевые кадры).

# Свойства анимации

Свойство animation является **сокращённым** для нескольких других:

- animation-name: название анимации;
- animation-duration: как долго длится анимация;
- animation-timing-function: как вычисляются промежуточные состояния;
- animation-delay: анимация начинается спустя некоторое время;
- animation-iteration-count: сколько раз должна выполняться анимация;
- animation-direction: должно движение идти в обратную сторону или нет;
- animation-fill-mode: какие стили применяются до начала анимации и после её завершения.

# Быстрый пример

- Для оживления кнопки загрузки, вы можете написать анимацию **подпрыгивания**:

```
@keyframes bouncing{
0% { bottom: 0; box-shadow: 0 0 5px rgba(0,0,0,0.5); }
100% { bottom: 50px; box-shadow: 0 50px 50px rgba(0,0,0,0.1); } }

 .loading-button {
    animation: bouncing 0.5s cubic-bezier(0.1,0.25,0.1,1) 0s infinite alternate both;
}
```

# Анимация

Сначала нужно написать реальную анимацию подпрыгивания с помощью @keyframes и назвать её bouncing. *Затем* вы можете использовать эту анимацию, применяя её к loading-button.

Используем **сокращенное** свойство animation и включаем все возможные варианты:

- animation-name: bouncing (совпадает с названием ключевых кадров)
- animation-duration: 0.5s (полсекунды)
- animation-timing-function: cubic-bezier(0.1,0.25,0.1,1)
- animation-delay: 0s (без задержки)
- animation-iteration-count: infinite (воспроизводится бесконечно)
- animation-direction: alternate (идёт назад и вперёд)
- animation-fill-mode: both

# @keyframes

Перед применением анимации к элементам HTML, вам требуется написать анимацию с помощью ключевых кадров.
Ключевые кадры — это каждый **промежуточный шаг** в анимации.
Они определяются с помощью процентов:

- 0% — первый шаг анимации;
- 50% — шаг на полпути в анимации;
- 100% — последний шаг.

Вы можете определить столько ключевых кадров, сколько хотите, вроде 33%, 4% или даже 29.86%. На практике вы будете писать только некоторые из них.
Каждый ключевой кадр является **правилом CSS**, это означает, что вы можете писать свойства CSS как обычно.
Чтобы определить анимацию, просто напишите ключевое слово @keyframes с его **названием**:

```
@keyframes around {
0% { left: 0; top: 0; }
25% { left: 240px; top: 0; }
50% { left: 240px; top: 140px; } 75% { left: 0; top: 140px; }
100% { left: 0; top: 0; } }

p { animation: around 4s linear infinite; }
```

*Обратите внимание, что начало 0% и конец 100% содержат **одинаковые правила** CSS. Это гарантирует, что анимация зациклится идеально. Поскольку счётчик итераций установлен как infinite, то анимация будет идти от 0% до 100%, а затем немедленно обратно к 0% и так бесконечно.*

# animation-name

Название анимации используется, по крайней мере, **дважды**:

- при **написании** анимации с помощью @keframes;
- при **использовании** анимации с помощью свойства animation-name (или через сокращённое свойство animation).

```
@keyframes whatever {
 /* ... */
}

.selector { animation-name: whatever; }
```

Подобно именам классов CSS, название анимации может включать в себя только:

- буквы (a-z);
- цифры (0-9);
- подчёркивание (_);
- дефис (-).

Название не может начинаться с цифры или с двух дефисов.

# animation-duration

Как и длительность перехода, длительность анимации может быть установлена в **секундах** (1s) или **миллисекундах** (200ms).

```
.selector { animation-duration: 0.5s; }
```

Значение по умолчанию равно 0s, что означает отсутствие анимации вообще.

# animation-timing-function

Подобно функциям времени для переходов, функции времени для анимации могут использовать **ключевые слова**, такие как linear, ease-out или могут быть определены с помощью произвольных **кривых Безье**.

.selector { animation-timing-function: ease-in-out; }

Значение по умолчанию: ease.

# animation-delay

Как и с задержкой перехода, задержка анимации может быть установлена в **секундах** (1s) или **миллисекундах** (200ms).
По умолчанию равно 0s, что означает отсутствие любой задержки.
Полезно использовать, когда включается несколько анимаций в **серии**.

```
.p { animation: bouncing 1s; }
```

# animation-iteration-count

По умолчанию, анимация воспроизводится только **один раз** (значение 1).
Вы можете установить три типа значений:

- целые числа, вроде 2 или 3;
- дробные числа, вроде 0.5, которые будут воспроизводить только половину анимации;
- ключевое слово infinite, которое будет повторять анимацию бесконечно.

```
.selector { animation-iteration-count: infinite; }
```

# animation-direction

- Свойство animation-direction определяет, *в каком порядке* читаются ключевые кадры.
- normal: начинается с 0%, заканчивается на 100%, начинается с 0% снова.
- reverse: начинается со 100%, заканчивается на 0%, начинается со 100% снова.
- alternate: начинается с 0%, идёт до 100%, возвращается на 0%.
- alternate-reverse: начинается со 100%, идёт до 0%, возвращается на 100%.

# animation-fill-mode

- Свойство animation-fill-mode определяет, что происходит *перед* началом анимации и *после* её завершения.
- При определении **ключевых кадров** можно указать **правила CSS**, которые будут применяться на разных шагах анимации. Теперь эти правила CSS могут *столкнуться* с теми, которые *уже применяются* к анимируемым элементам.
- animation-fill-mode позволяет сообщить браузеру, если *стили анимации* **также** должны применяться **за пределами анимации**.
- Давайте представим себе **кнопку**, которая является:
- **красной** по умолчанию;
- становится **синей** в начале анимации;
- и в итоге **зелёной**, когда анимация завершена.

| animation-fill-mode | До анимации | Начало анимации | Конец анимации | После анимации |
|---|---|---|---|---|
| none | По умолчанию | Начало | Конец | По умолчанию |
| forwards | По умолчанию | Начало | Конец | Конец |
| backwards | Начало | Начало | Конец | По умолчанию |
| both | Начало | Начало | Конец | Конец |

# Задание

Создать анимацию снеговика:

1. Начальные элементы (три круглые кнопки) находятся в разных сторонах окна браузера.

2. Еще один элемент (ведро или морковка)  тоже.

3. При наведение на элементы, они должны плавно передвигаться к середине экрана, создавая «собранный» снеговик.