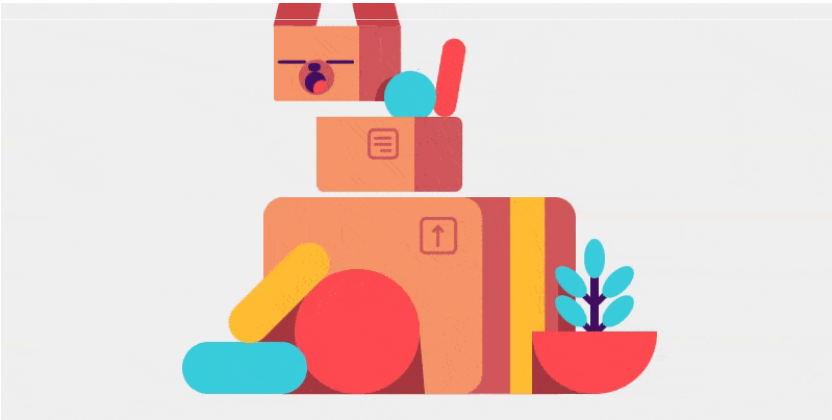
Unit 2

ANIMATION

CSS Animations and CSS Transitions

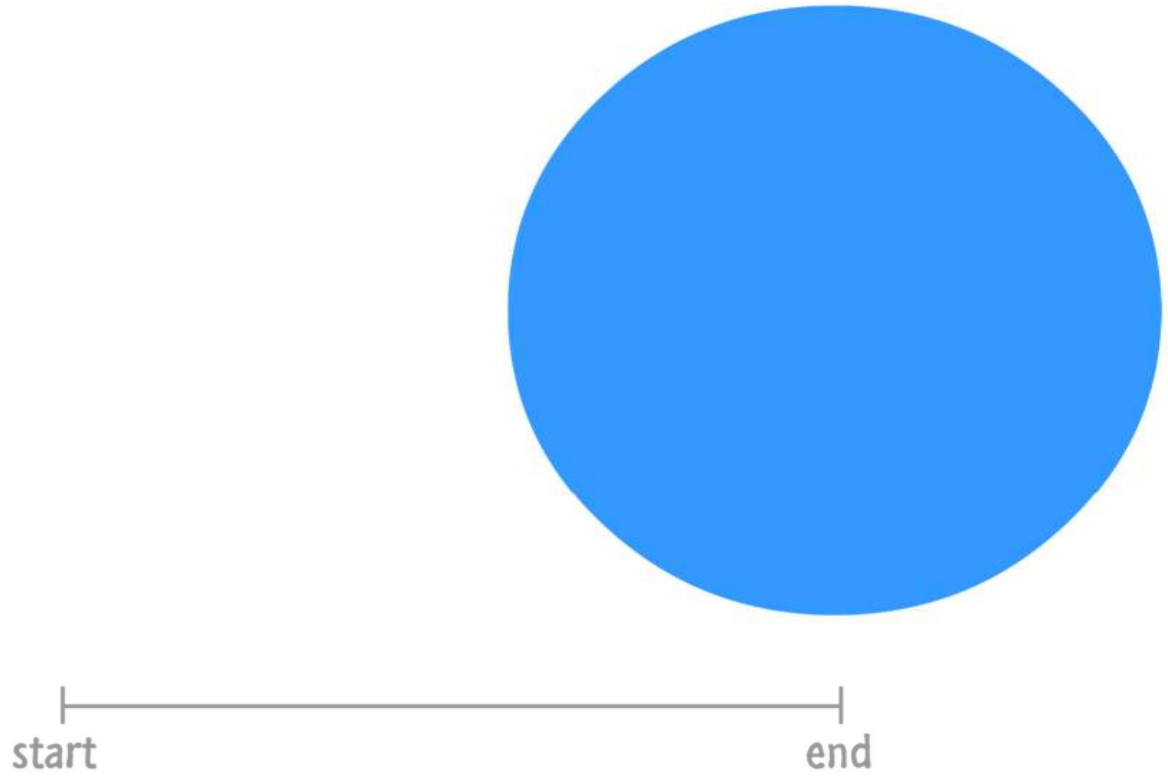


What is animation

- An animation is nothing more than a visualization of change.
- It is the process of creating a scene through the rapid display of pictures and motions.
- animations help to make applications
  - easier to navigate,
  - content be more presentable,
  - feel more alive and fun.

The Start and End States

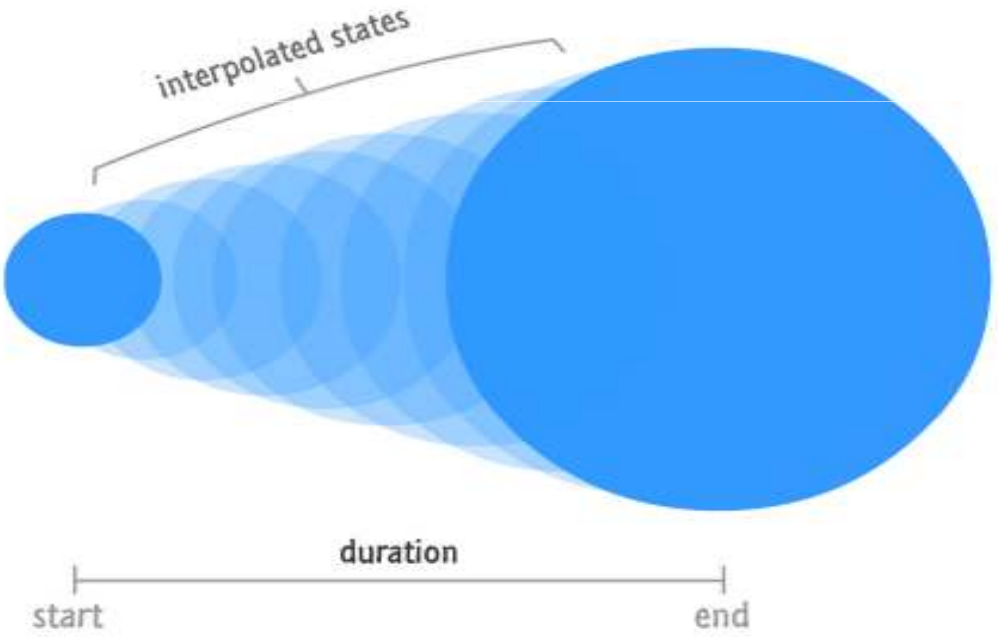
- The start and end states are the reference points so that we can compare what has changed during an animation.
- Example: Consider the blue circle at the start state of animation



- Now, at the end state of blue circle animation, One change is the position. The blue circle starts off on the left side of the page. In the end, it is located on the right hand side.
- Another change is the size. The blue circle became several times larger in the end state compared to its initial size at the beginning.

Interpolation

- Interpolation means the intermediate states are generated between the start and end states over a period of time.
- Hence we need to specify the start state, end state and duration for which the the transition between the two states needs to occur



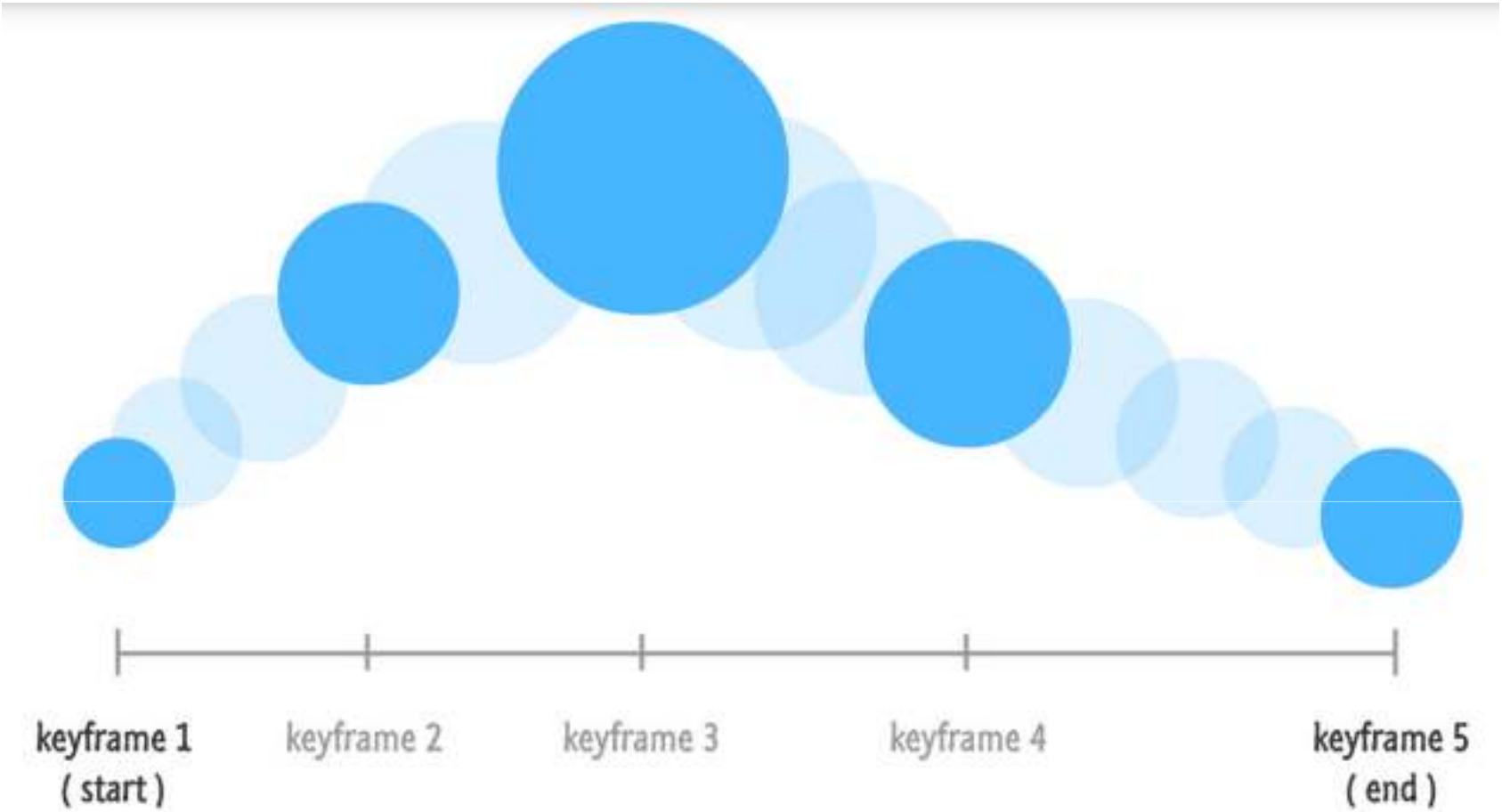
Animations in HTML

- In HTML there are 3 flavours of animation

1) **CSS Animations (aka Keyframe Animations)** → It is a traditional class of animations where we define the start state, end state, and intermediate states called keyframes
2) **CSS Transitions** → It is a class of animations where we only define the start state, end state, and duration.
3) **JavaScript Animations** → It is a class of animations where we define the start state, end state, and duration and the intermediate states (interpolation)

1. CSS Animations

- CSS animations are the traditional animations which require the intermediate states called keyframes to be specified along with start and end states.
- Using this we can change as many CSS properties we want, as many times as we want.
- Keyframes hold what styles the element will have at certain times.
- example: making things move, having things fade in and out, seeing things change color, etc


keyframe 1
(start)
keyframe 2
keyframe 3
keyframe 4
keyframe 5
(end)

How to begin CSS animation

- The first step is to set the **animation property** for the element
- The second step is to **define the keyframes** that specify exactly what gets animated.
- i.e. we must specify CSS styles inside the @keyframes rule, so that the animation will gradually change from the current style to the new style at certain times.
- Example 1:

```
#bigcloud {
    animation: bobble 2s infinite;
    margin-left: 100px;
    margin-top: 15px; }
```

```
@keyframes bobble
{
0% {
        transform: translate3d(50px, 40px, 0px);
         animation-timing-function: ease-in;
     }
 50% {       transform: translate3d(50px, 50px, 0px);
             animation-timing-function: ease-out;
       }
 100% {       transform: translate3d(50px, 40px, 0px);    }
}
```

Example 2:

```
/* Set the animation property to bind it to an Html element*/
    div {
      width: 100px;
      height: 100px;
      background-color: red;
      animation-name: example;
      animation-duration: 4s;
    }

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

- The animation property is responsible for setting your animation up. We have to specify atleast 3 things :
  1. The name of our animation
  2. The duration
  3. The number of times your animation will loop

  Example : **animation: bobble 2s infinite;**

- Further , the @keyframes declaration is followed by the name of our animation. On the inside, it contains style rules (aka the actual keyframes) whose selectors are either percentage values or the keywords from and to . These values will get applied when the rule becomes active.
- The name we give to our @keyframes rule acts an identifier the animation property uses to know where the keyframes are.

Css properties Animation

- animation
- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode

- **animation-duration** → property defines how long an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).
- **animation-delay**→ property specifies a delay for the start of an animation.
- **animation-iteration-count** → property specifies the number of times an animation should run.
- **animation-play-state** → property is used to toggle between the paused or play (running state) in middle of the animation

- **animation-direction** → property specifies whether an animation should be played forwards, backwards or in alternate cycles.
- The animation-direction property can have the following values:
  - **normal** - The animation is played as normal (forwards). This is default
  - **reverse** - The animation is played in reverse direction (backwards)
  - **alternate** - The animation is played forwards first, then backwards
  - **alternate-reverse** - The animation is played backwards first, then forwards

- **animation-timing-function** → property specifies the speed curve of the animation.
- The animation-timing-function property can have the following values:
  - **ease** - Specifies an animation with a slow start, then fast, then end slowly (this is default)
  - **linear** - Specifies an animation with the same speed from start to end
  - **ease-in** - Specifies an animation with a slow start
  - **ease-out** - Specifies an animation with a slow end
  - **ease-in-out** - Specifies an animation with a slow start and end
  - **cubic-bezier(n,n,n,n)** - Lets you define your own values in a cubic-bezier function

- **animation-fill-mode** → property specifies a style for the element when the animation is not playing (before it starts, after it ends, or both).
- i.e. CSS animations do not affect the element before the first keyframe is played or after the last keyframe is played.
- It has following property values:
  - **none** → Default value. Animation will not apply any styles to the element before or after it is executing
  - **forwards** → The element will retain the style values that is set by the last keyframe
  - **backwards**→ The element will get the style values that is set by the first keyframe and during the delay period
  - **both** → The animation will follow the rules for both forwards and backwards

The Animation Shorthand :

- Animation shorthand properties is a compact way of specifying the animation properties.
- Here all the properties are specified in the animation property itself.
- Syntax:

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

Example:

animation: bobble 2s ease-in infinite;

animation: bobble 4s ease-in 2s infinite alternate;

Animation Longhand properties → specifying the animation properties seperately

- Example:

```
div {
    animation-name: bobble;
    animation-duration: 2s;
    animation-timing-function: ease-in;
    animation-iteration-count: infinite;
    }
```

Declaring Multiple Animations:

- To declare multiple animations in the same animation property, simply comma separate each of your animations in the shorthand declaration.
- example:

```
#multiStyle {
animation: first  2s infinite,
                second 1s infinite,
                third 5s infinite;
               }
```

2. CSS Transitions

- When interacting with UIs, we see reactions to the things we are doing. For animating these kinds of situations, we have **CSS Transitions**.
- **Transitions animate the property changes smoothly over a given duration.**
- Without Transitions , the changes in Css properties is really sudden or abrupt
- They don’t involve any keyframes
- Examples of such reactions include
  - a link underlining when you hover it,
  - a menu flying in when you tap or click on a button,
  - a text element getting bigger when it has focus,

Adding a transition

- To add or create a transition, we need to specify at least two things:
  1. The CSS property we want our transition to listen for changes on.
  2. How long the transition will run or duration
- Example:

```
#hexagon {   transition: transform  0.1s;  }
#hexagon:hover
{
transform: scale3d(1.2, 1.2, 1) rotate(45deg);
}
```

Css transition properties

Property	Description
transition	A shorthand property for setting the four transition properties into a single property
transition-delay	Specifies a delay (in seconds) for the transition effect
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-property	Specifies the name of the CSS property the transition effect is for
transition-timing-function	Specifies the speed curve of the transition effect

- **The transition-property** → specifies the name of the CSS property the transition effect is for (the transition effect will start when the specified CSS property changes).

  syntax:

  transition-property: property | none | all |initial | inherit;

  Example:

```
div {
  transition-property: width, height;
}
```

- The **transition-timing-function** can have the following values:
  - **ease** - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
  - **linear** - specifies a transition effect with the same speed from start to end
  - **ease-in** - specifies a transition effect with a slow start
  - **ease-out** - specifies a transition effect with a slow end
  - **ease-in-out** - specifies a transition effect with a slow start and end
  - **cubic-bezier(n,n,n,n)** - lets you define your own values in a cubic-bezier function

Multiple Css Transitions

- To apply multiple CSS Transitions on an element , we just have to specify the transitions in a **shorthand** form by adding the property, time duration, and timing function (whichever properties are required) and **separate them with commas**.
- Example:

  transition: color 1s ease-out, padding-top 1s ease-out, padding-bottom 1s ease-out, font-size 2s ease-out;

Longhand property

- Using longhand properties, we can set different transition properties separately one by one.
- example:

```
#hexagon
{
    transition-property: transform;
    transition-duration:  0.1s;
}
```