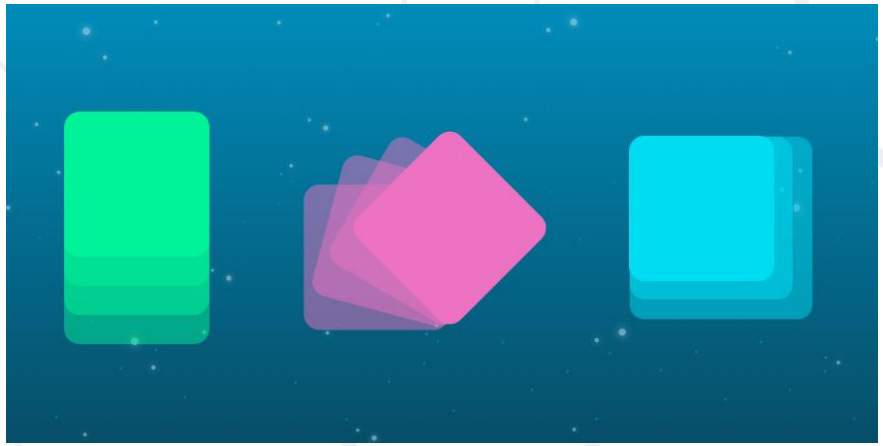
Transformations, Transitions and Animations



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.org

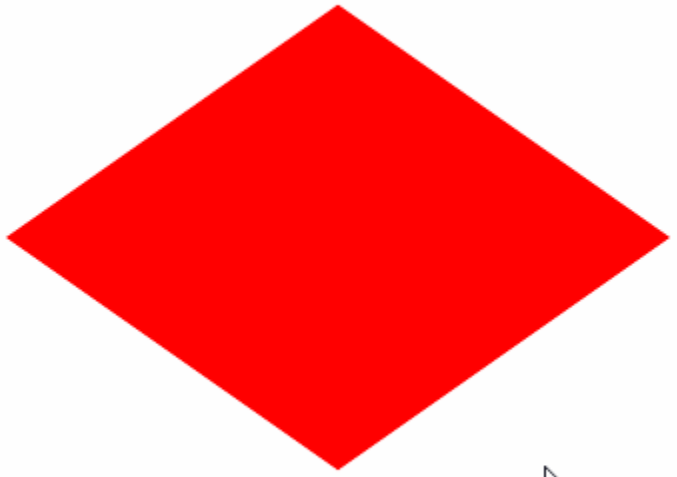
Table of Contents

Have a Question?

Software University

sli.do

#front-end



Transformations

Rotate, Scale, Move, Skew, Etc., Elements

CSS Transformations

- The `transform` property applies a **2D** or **3D** **transformation** to an element
- This property allows you to:
  - Rotate
  - Scale
  - Move
  - Skew

```
div.a {
    transform: rotate(20deg);
}
```

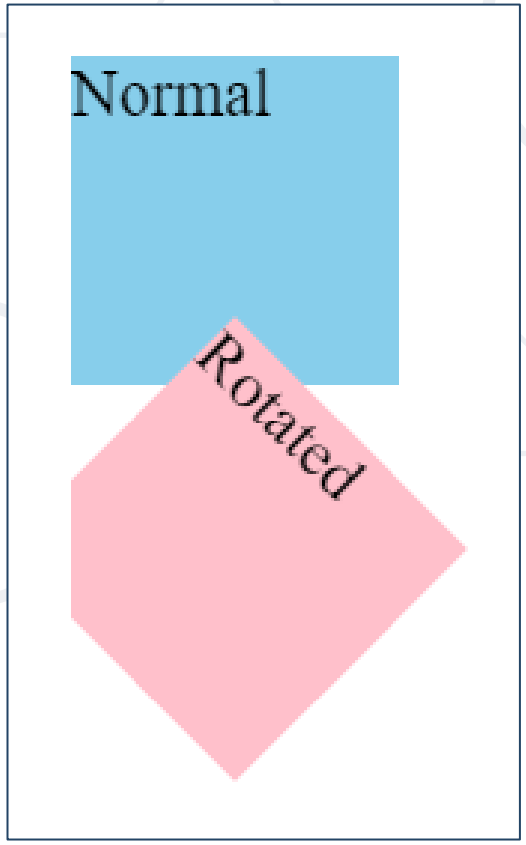
CSS Transformations - Rotate

- The `rotate()` CSS function defines a transformation that rotates an element around a fixed point

```
<div>Normal</div>
<div class="rotated">Rotated</div>
```

```
div {
    width: 80px;
    height: 80px;
    background-color: skyblue;
}

.rotated {
    transform: rotate(45deg);
    background-color: pink;
}
```



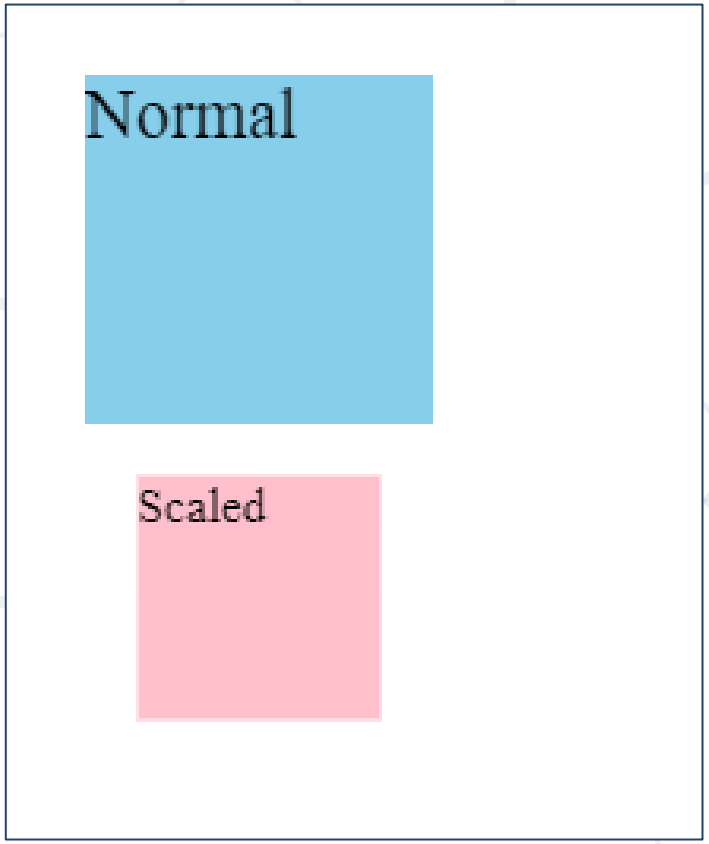
CSS Transformations - Scale

- The `scale()` CSS function defines a transformation that resizes an element

```
<div>Normal</div>
<div class="scaled">Scaled</div>
```

```
div {
    width: 80px;
    height: 80px;
    background-color: skyblue;
}

.scaled {
    transform: scale(0.7);
    background-color: pink;
}
```

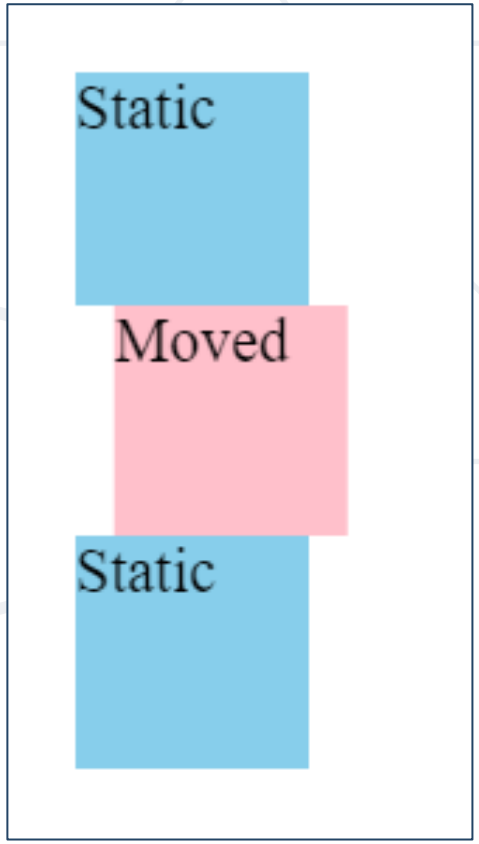


CSS Transformations - Translate

- The `translate()` CSS function repositions an element in the horizontal and/or vertical directions

```
<div>Static</div>
<div class="moved">Moved</div>
<div>Static</div>
```

```
div {
    width: 60px;
    height: 60px;
    background-color: skyblue;
}
.moved {
    transform: translate(10px);
    background-color: pink;
}
```



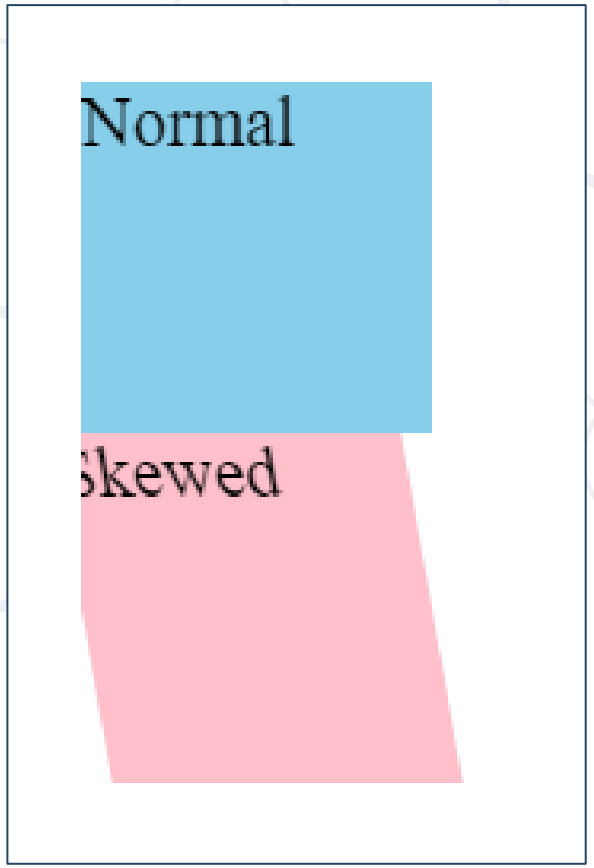
CSS Transformations - Translate

- The `skew()` function defines a transformation that skews an element

```
<div>Normal</div>
<div class="skewed">Skewed</div>
```

```
div {
    width: 80px;
    height: 80px;
    background-color: skyblue;
}

.skewed {
    transform: skew(10deg);
    background-color: pink;
}
```



Transformation Properties

- `none` - Defines that there should be **no transformation**
- `matrix(n,n,n,n,n,n)` - Defines a **2D transformation**, using a matrix of six values
- `translate(x,y)` - Defines a **2D translation**
- `translateX(x)` - Defines a translation, using only the value for the **X-axis**
- `translateY(y)`- Defines a translation, using only the value for the **Y-axis**

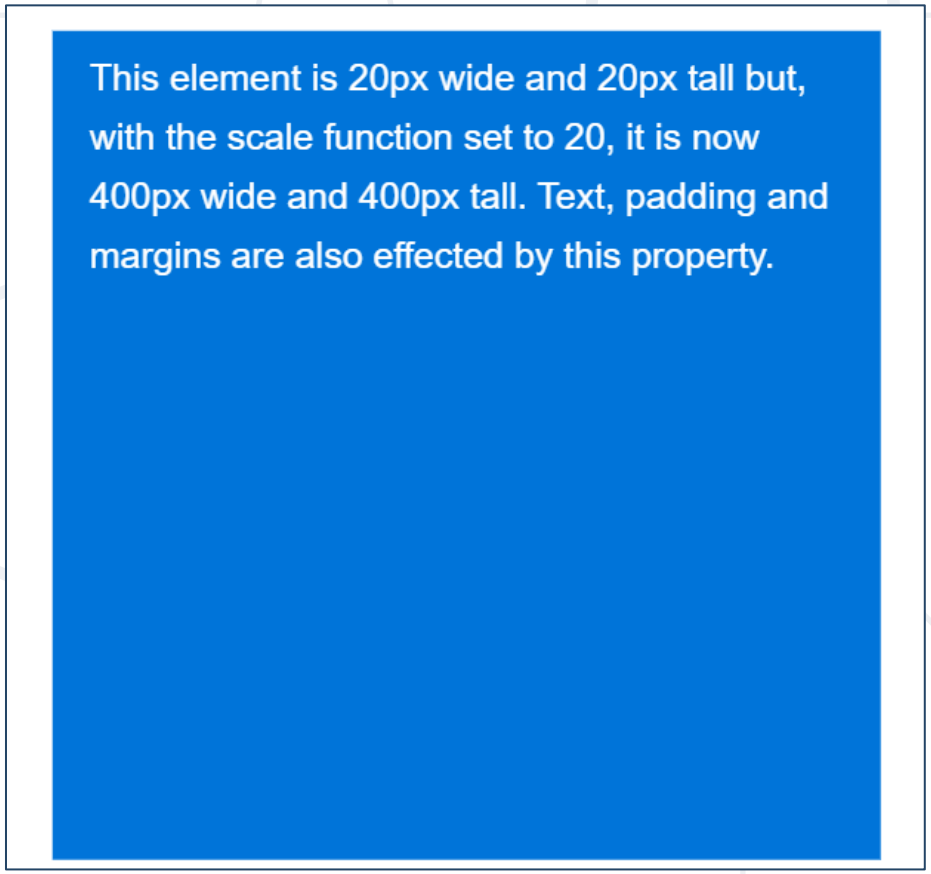
Transformation Properties

- `scale(x,y)` - Defines a 2D **scale** transformation
- `rotate(angle)` - Defines a 2D **rotation**, the angle is specified in the parameter
- `skew(x-angle,y-angle)` - Defines a 2D **skew** transformation along the X- and the Y-axisa
- `perspective(n)` - Defines a **perspective** view for a 3D transformed element
- `initial` - Sets this property to its **default value**
- `inherit` - Inherits this property from its **parent element**

Transform Example

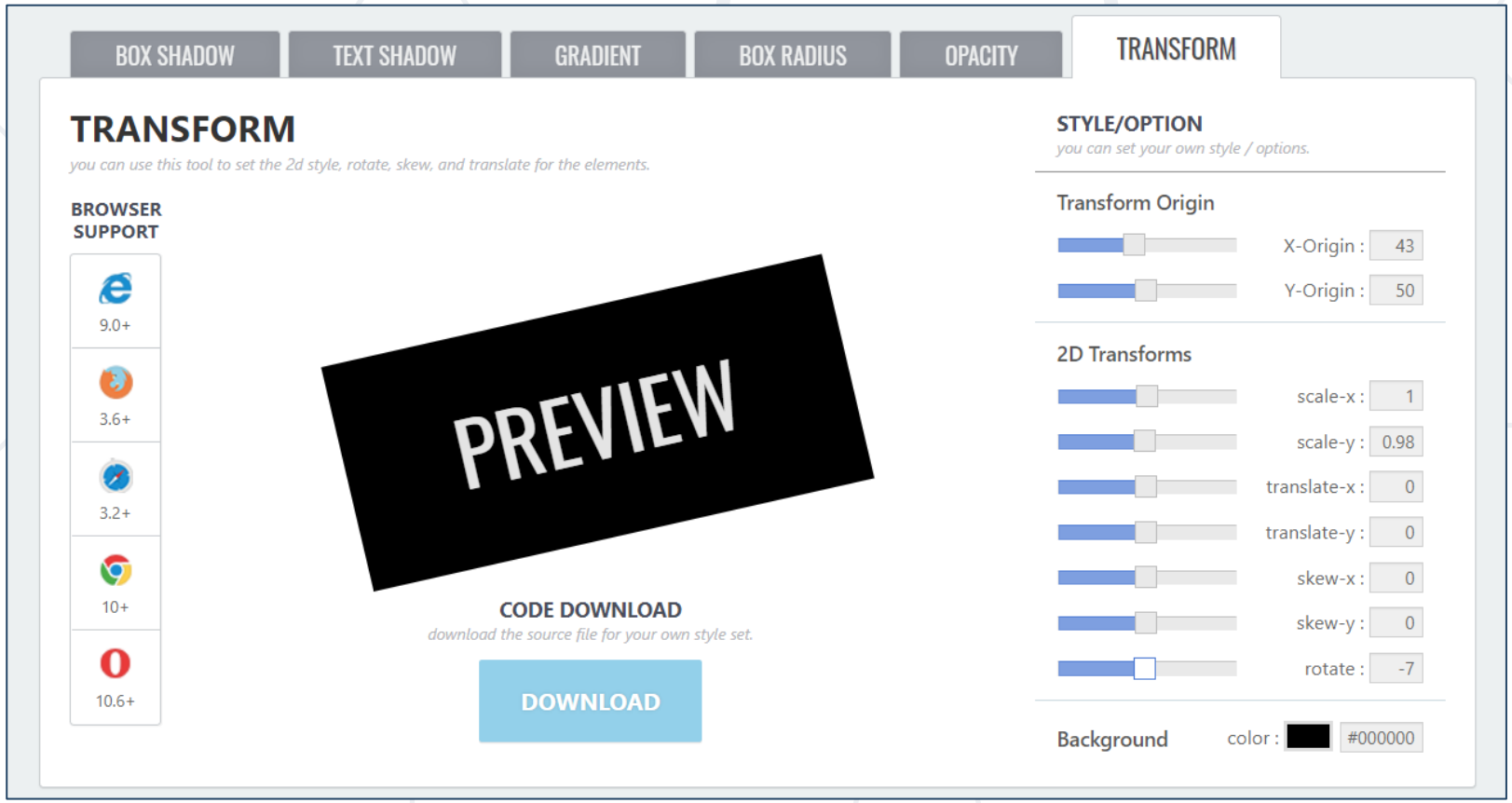
- Even with a declared **height** and **width**, this element will now be **scaled** to **twenty** times its original size

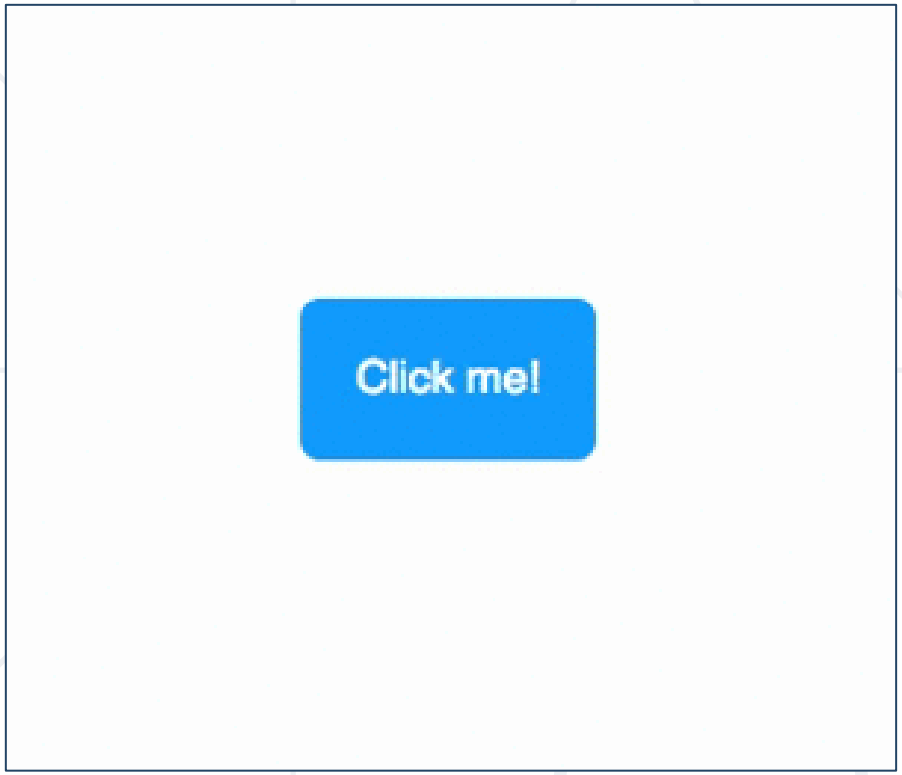
```
.element {
    width: 20px;
    height: 20px;
    transform: scale(20);
}
```



Transform Generator

- A Transform CSS Generator that helps you quickly generate transform CSS declarations



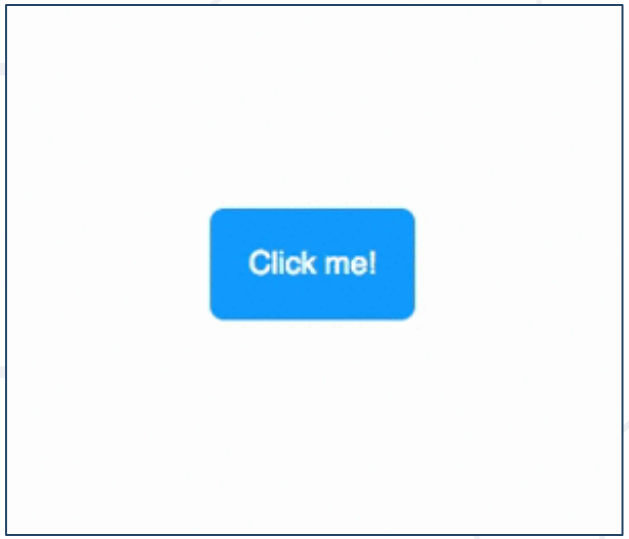


CSS Transitions

Change Property Values Smoothly, Over a Given Duration

CSS Transitions

- Allows you to change property values **smoothly**, over a **given duration**
  - Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time



CSS Transitions

- If you change the color of an element from white to black, usually the change is **instantaneous**
  - With CSS transitions enabled, changes occur at **time intervals** that follow an **acceleration curve**, all of which can be **customized**

Transition Properties

- `transition`
- `transition-delay`
- `transition-duration`
- `transition-property`
- `transition-timing-function`
- Transitions - **Shorthand**

```
/* property name | duration | timing function | delay */
transition: color 4s ease-in-out 1s;
```

Transitions - Timing Functions

- Mathematical function that describes **how fast** one-dimensional values **change during** animations
  - This lets you vary the **animation's speed** over the course of its duration
- "**Smooth**" timing functions are often called **easing functions**
  - They correlate a time ratio to an output ratio, both expressed as **s**
    - For these values, 0.0 represents the **initial state**, and 1.0 represents the **final state**

Ease Out

- Easing out causes the animation to **start more quickly** than linear ones, and it also has **deceleration** at the end
- Good for user interface work
  - The fast start gives your animations a feeling of responsiveness, while still allowing for a natural slowdown at the end

```
transition: transform 500ms ease-out;
```

ease-in

Ease In

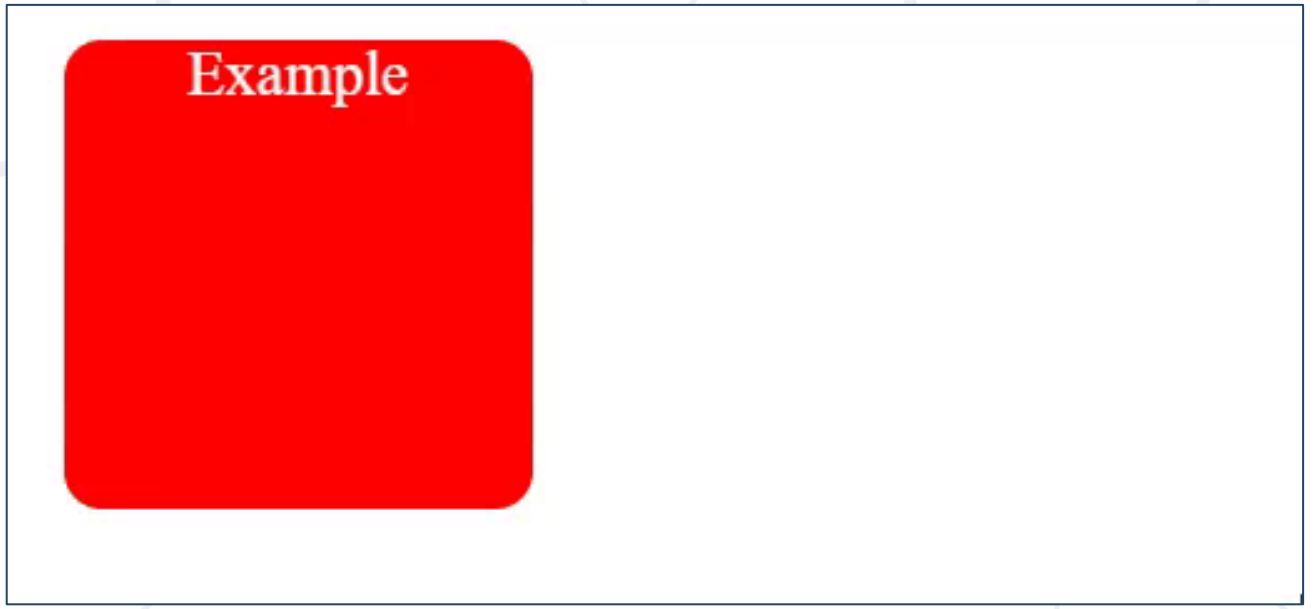
- Start **slowly** and **end fast**
- Have the detrimental effect of feeling sluggish when starting, which negatively impacts the perception of responsiveness in your site

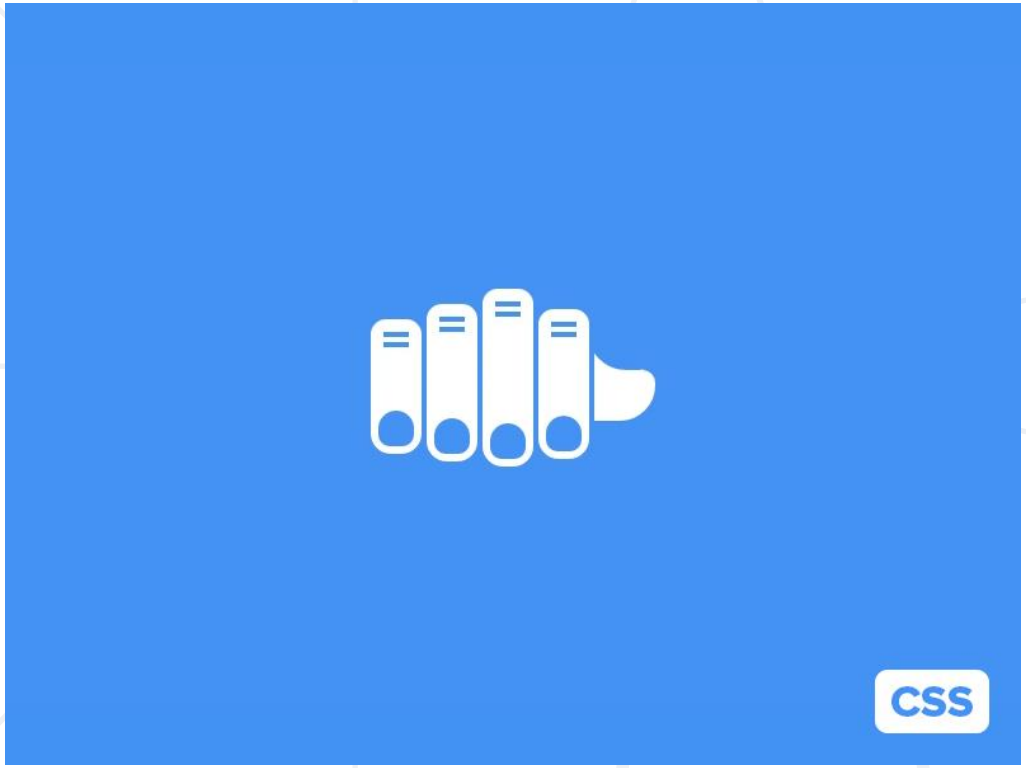
```
transition: transform 500ms ease-in;
```

ease-in

Problem: Box Change

- Update the following box properties with transition `ease` effect:
  - `width;`
  - `height;`
  - `background-color;`
  - `font-size;`
  - `left;`
  - `top;`
  - `color;`





Animations

Animate the Values of CSS Properties

CSS Animations

- A module that lets you animate the values of CSS properties over time, using **keyframes**
- The behavior of these **keyframe animations** can be controlled by specifying their:
  - Timing function
  - Duration
  - Number of repetitions
  - Other attributes

Keyframes

- The `@keyframes` CSS `at-`rule controls the intermediate steps in a CSS animation sequence by defining styles for keyframes along the animation sequence
  - This gives **more control** over the intermediate steps of the animation sequence than transitions

Keyframes - Example

```
@keyframes slideIn {
    from {
        transform: translateX(-20rem);
        opacity: 0;
    }

    to {
        transform: translateX(0);
        opacity: 1;
    }
}
```

Keyframes - Example

```
@keyframes slideIn {
    0% {
        transform: translateX(-20rem);
        opacity: 0;
    }

    50% {
        transform: translateX(10rem);
    }

    100% {
        transform: translateX(0);
        opacity: 1;
    }
}
```

Animations – Properties

- `animation-delay` - Configures the delay between the time the element is loaded and the beginning of the animation sequence
- `animation-direction` - Configures whether the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself
- `animation-duration` - Configures the length of time that an animation should take to complete one cycle

Animations – Properties

- `animation-iteration-count` - Configures the number of times the animation should repeat
  - You can specify infinite to repeat the animation indefinitely
- `animation-name` - Specifies the name of the `@keyframes at-` rule describing the animation's keyframes
- `animation-play-state` - Lets you pause and resume the animation sequence

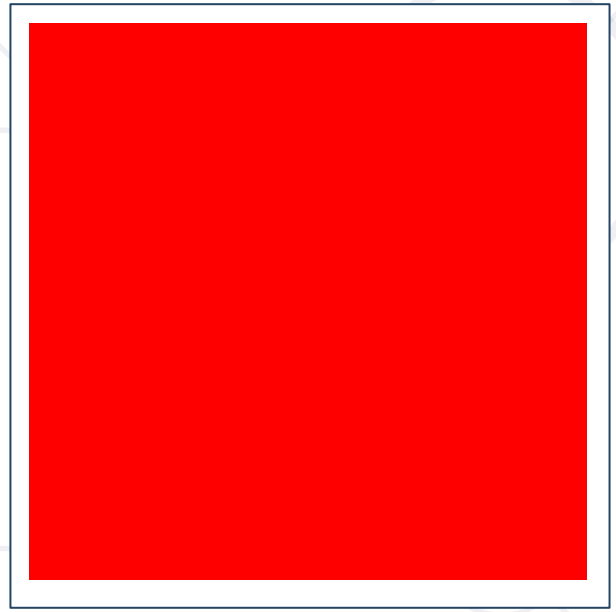
Animations – Properties

- `animation-timing-function` - Configures the timing of the animation
  - That is, how the animation transitions through keyframes, by establishing acceleration curves
- `animation-fill-mode` - Configures what values are applied by the animation before and after it is executing

Animation - Example

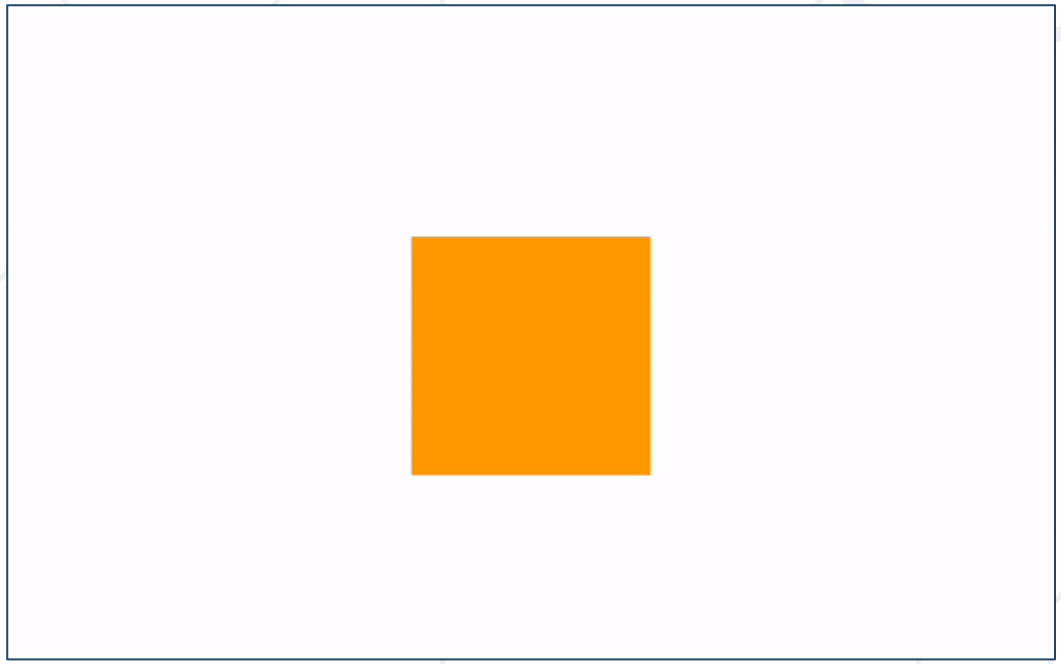
```
div {
    width: 100px;
    height: 100px;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
}

@keyframes example {
    from {
        background-color: red;
    }
    to {
        background-color: yellow;
    }
}
```



Problem: Circle to Box Animation

- Create a **small red circle**
- **Center** the circle in the viewport
- **Animate** that circle to a square and back to circle



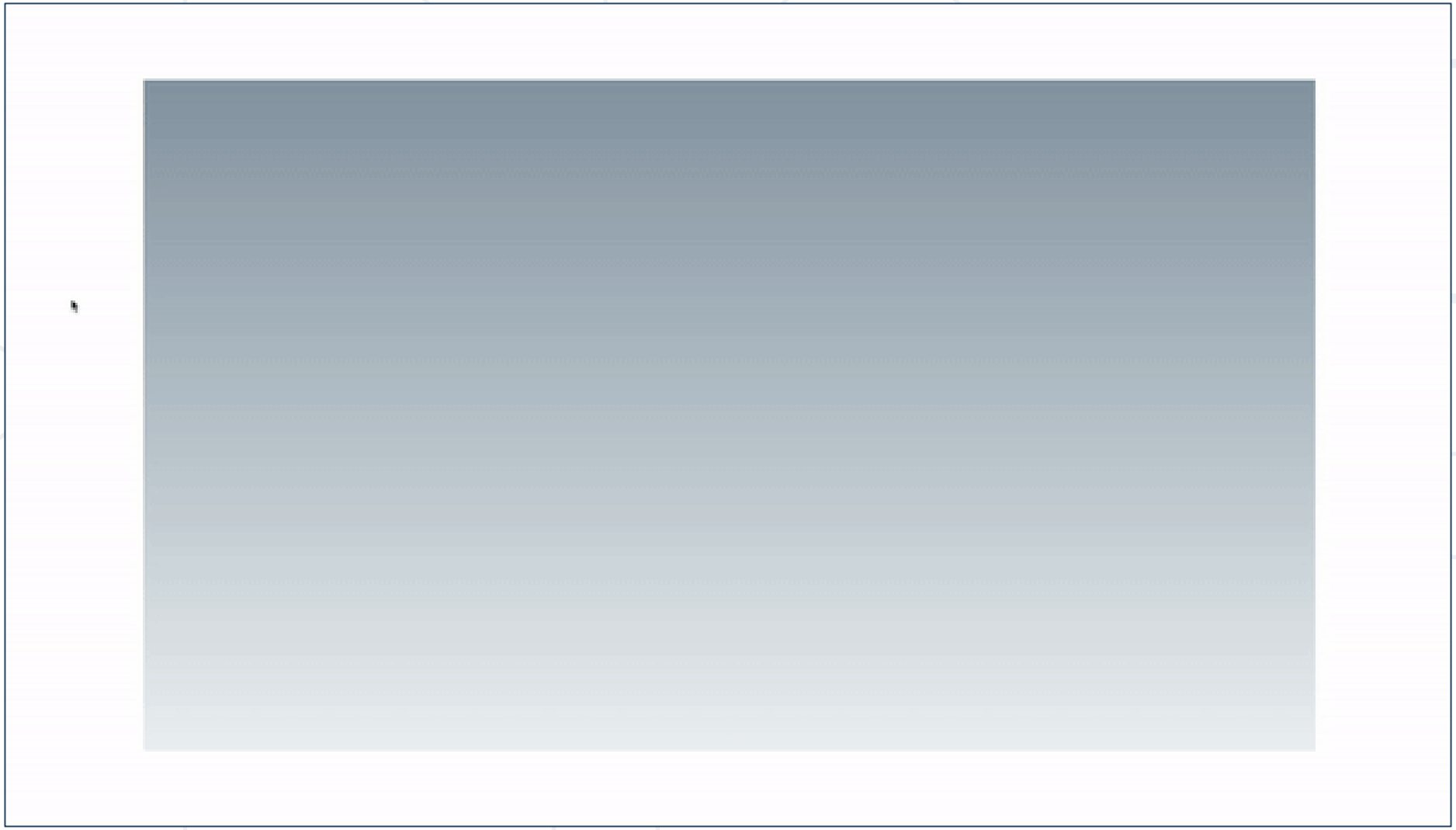
Problem: Fancy List Animation

- Create a **list of items** with icons
- Animate the showing of **each element**
- Use **`animation-delay`** to make the appearance of the elements staggered

Problem: Animating Gradients

- Create an empty `div` element
- Use `:before` and `:after` pseudo elements with position absolute to make them as big as their container
- Add different gradients on the `:before` and `:after` elements
- Animate between the different gradients by using `opacity` on the top most element

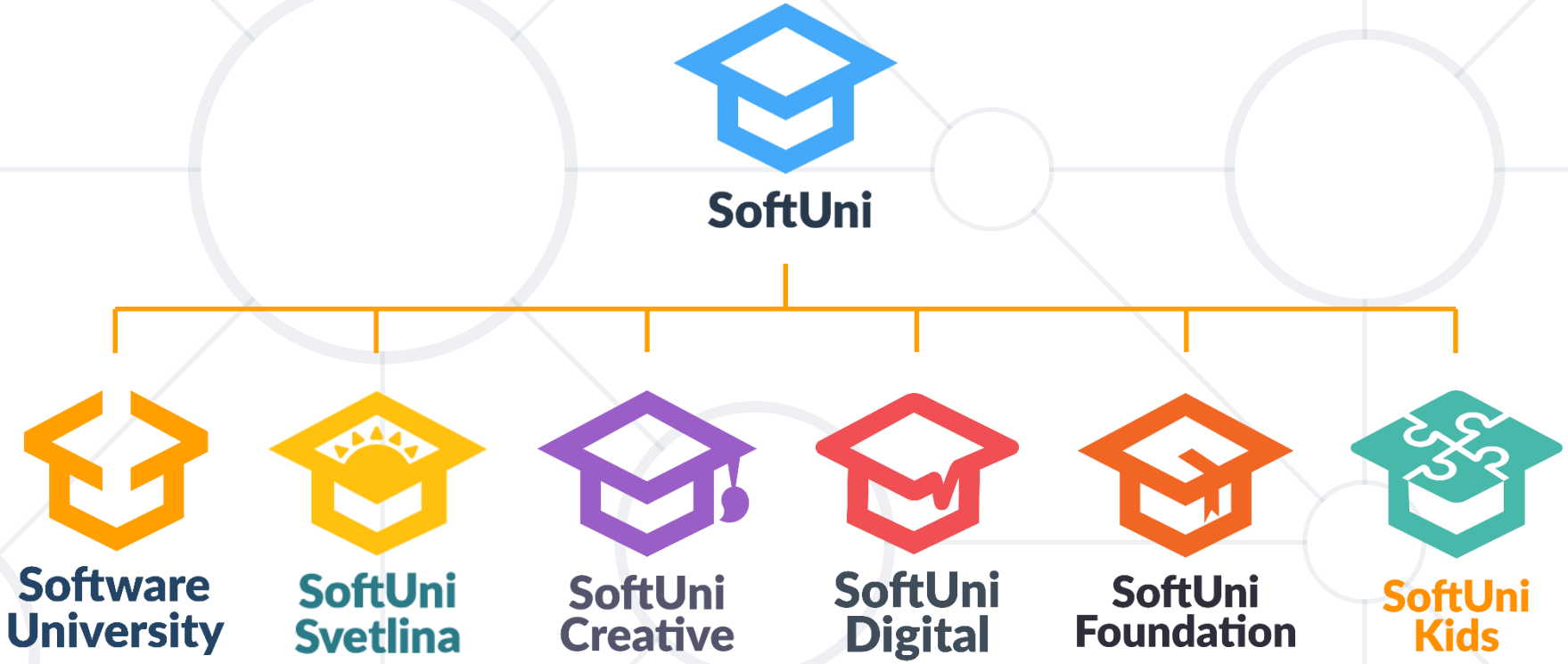
Problem: Animating Gradients



Summary

- **Transformations**
  - Transform Properties
- **Transitions**
  - Transition Properties
  - Timing Functions
- **Animations**
  - Animation Properties
  - Keyframes

Questions?



© SoftUni – https://softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

SoftUni Diamond Partners

Software University

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
  - Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg