Advanced CSS

Week 2 Session2

Contents of This session

- CSS 3
  - Animation
  - Transition
  - Transform

CSS Animation

- The CSS Animations module allows authors to create real, honest-to-goodness keyframe animation
- keyframe animation allows you to explicitly specify other states at points along the way
- Those “points along the way” are established by *keyframes* that define the beginning or end of a segment of animation

The Building Blocks of Animations

- **Keyframes** - define the stages and styles of the animation.
- **Animation Properties** - assign the @keyframes to a specific CSS element and define *how* it is animated.

Syntax:

```
@keyframes animation-name {
    keyframe { property: value; }
    keyframe { property: value; }
}
```

Building Block #1: Keyframes

- **Name of the animation:** A name that describes the animation, for example, bounceIn.
- **Stages of the animation:** Each stage of the animation is represented as a percentage. 0% represents the beginning state of the animation. 100% represents the ending state of the animation. Multiple intermediate states can be added in between.
- **CSS Properties:** The CSS properties defined for each stage of the animation timeline.

Example 1

```
/* The animation code */
@keyframes exampleA {
    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: exampleA;
    animation-duration: 4s;
}
```

CSS Animation: keyframes

- Example B of Keyframes

```
@keyframes exampleB {
    0% { background-color: red; }
    20% { background-color: orange; }
    40% { background-color: yellow; }
    60% { background-color: green; }
    80% { background-color: blue; }
    100% { background-color: purple; }
}
```

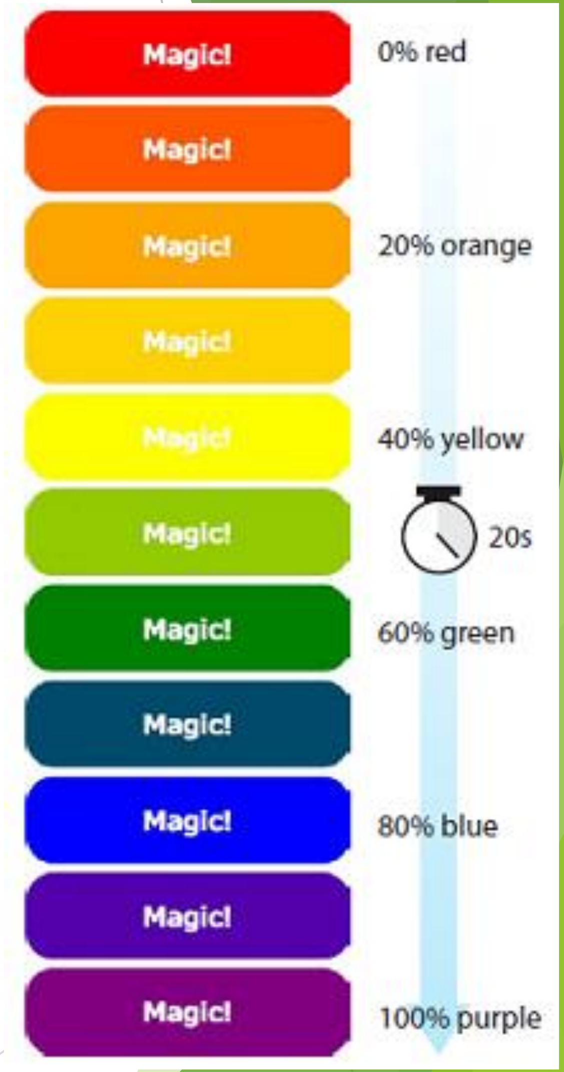
simplistic set of keyframes that changes the background color of an element over time

CSS Animation: keyframes

- Example of Keyframes

```
@keyframes colors {
    0% { background-color: red; }
    20% { background-color: orange; }
    40% { background-color: yellow; }
    60% { background-color: green; }
    80% { background-color: blue; }
    100% { background-color: purple; }
}
```

simplistic set of keyframes that changes the background color of an element over time



CSS Animation: Applying Animation

- Now we can apply this animation sequence to an element or multiple elements in the document using a collection of animation properties
- We can make some decisions about the animation we want to apply:
  - Which animation to use (**animation-name**)
  - How long it should take (**animation-duration**)
  - The manner in which it should accelerate (**animation-timing-function**)
  - Whether to pause before it starts (**animation-delay**)

CSS Animation: Applying Animation (con.)

- There are a few other animation-specific properties as well:
  - How many times it should repeat (**animation-iteration-count**).
  - Whether it plays forward, in reverse, or alternates back and forth **(animation-direction)**
  - Whether it should be running or paused. The play-state can be toggled on and off with JavaScript or on hover (**animation-play-state**).
  - Whether to override defaults that prevent properties from applying outside runtime (**animation-fill-mode**)

CSS Animation: Applying Animation (con.)

- Here is the resulting rule for the animated element

```
#magic {
    …
    animation-name: colors;
    animation-duration: 5s;
    animation-iteration-count: infinite;
    animation-direction: forward;
}
```

- Let have a look at our demos

CSS Transitions

- Picture in your mind, if you will, a link in a navigation menu that changes from blue to red when the mouse hovers over it.
- The background is blue… mouse passes over it…BAM! Red!
- Now imagine putting your mouse over the link and the background gradually changes from blue to red, passing through several shades of purple on the way. It’s smoooooth.
- That’s what *CSS Transitions* do.

CSS Transitions

- When applying a transition, there are a few decisions to make, each of which is set with a CSS property:
  - Which CSS property to change (**transition-property**)
  - How long it should take (**transition-duration**)
  - The manner in which the transition accelerates (**transition-timingfunction**)
  - Whether there should be a pause before it starts (**transition-delay**)

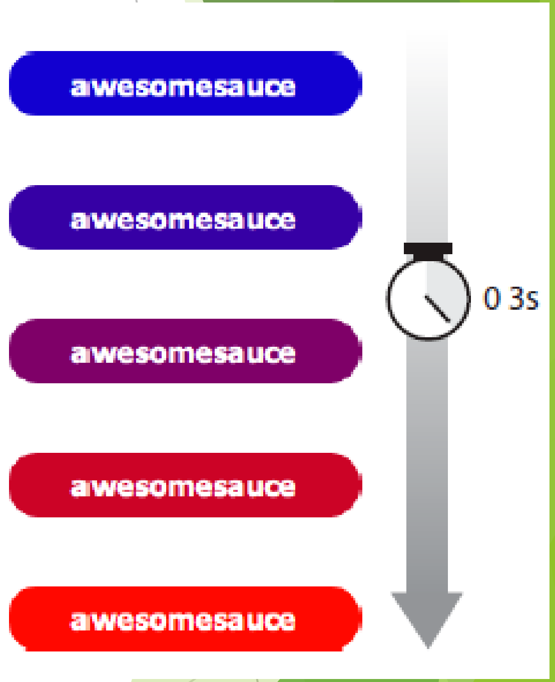


Illustration of Transition

CSS Transitions : transition-property

- specifies the name of the CSS property the transition effect is for
- **transition-property**

  Values: property-name | all | none

  Default: all
- Example

  transition-property: width;

  This means that you want to apply a transition to width property

CSS Transitions : transition-duration

- specifies how many seconds (s) or milliseconds (ms) a transition effect takes to complete
- **transition-duration**

  Values: time

  Default: 0s
- Example

  transition-duration: 5s;

  5s to complete the transition

Example 1

```
div {
    width: 100px;
    height: 100px;
    background: red;
    -webkit-transition-property: width; /* Safari */
    -webkit-transition-duration: 5s; /* Safari */
    transition-property: width;
    transition-duration: 5s;
}

div:hover {
    width: 300px;
}
```

Example 1-version 2

```
div {
    width: 100px;
    height: 100px;
    background: red;
    -webkit-transition: width 5s; /* For Safari
3.1 to 6.0 */
    transition: width 5s;
}

div:hover {
    width: 300px;
}
```

CSS Transitions
: transition-timing-function

- specifies the speed curve of the transition effect
- **transition-timing-function**

  Values: **ease | linear | ease-in | ease-out |**

  ease-in-out | step-start | step-end | steps

  Default: ease
- Example

  transition-timing-function: ease-out

This means the transition will starts out fast, then slows down

CSS Transitions
: transition-delay

- specifies when the transition effect will start
- **transition-delay**

  Values: time

  Default: 0s
- Example

  transition-delay: 2s

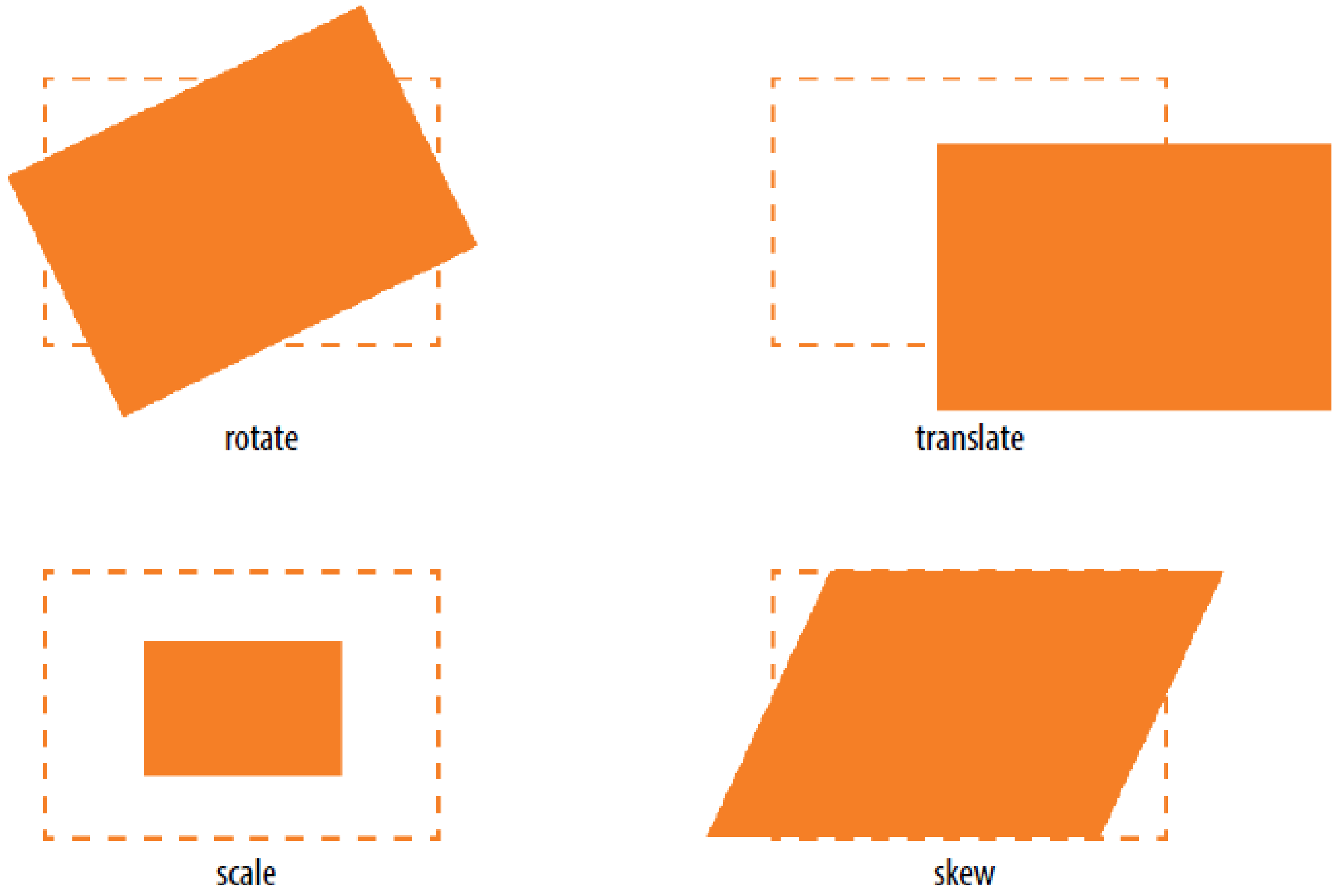
This means the transition will starts after 2 seconds

- Now, The demo time !!

CSS Transforms

- The CSS3 Transforms module gives authors a way to rotate, relocate, resize, and skew HTML elements in both two- and three-dimensional space
- We focus on the more straightforward 2-D varieties because they have more practical use

CSS Transforms



Four types of transformation

CSS Transforms

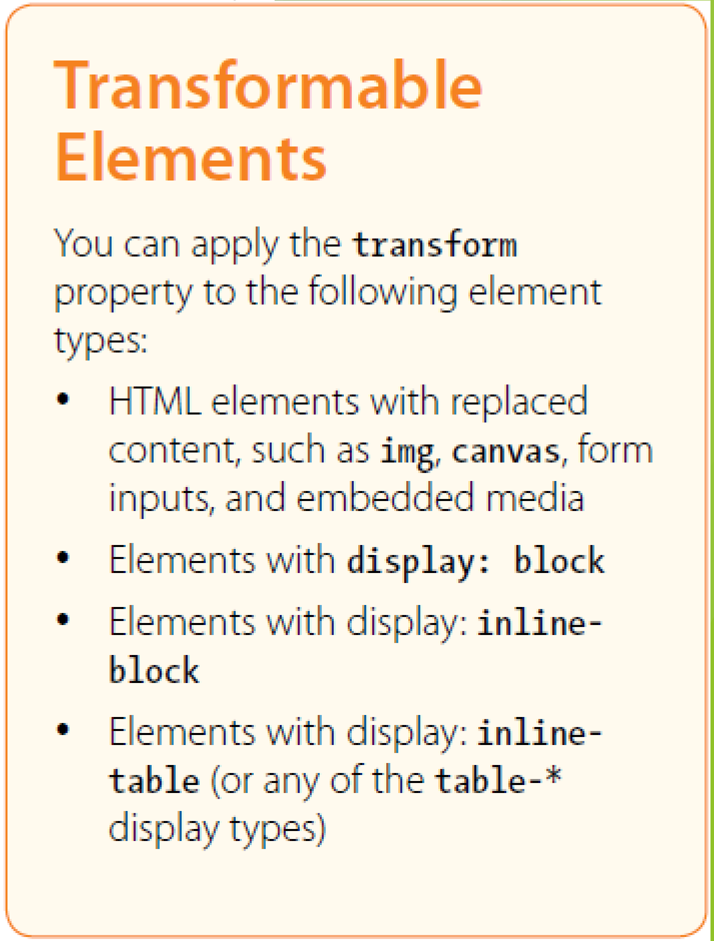
- allow you to translate, rotate, scale, and skew elements
- **transform**

  Values: transform function(s) **| none**

  Default: none
- Example

  `transform: translate(50px,100px);`

  We call a translate function to perform a transformation
- More details on transform functions in the following slides



Transformable Elements

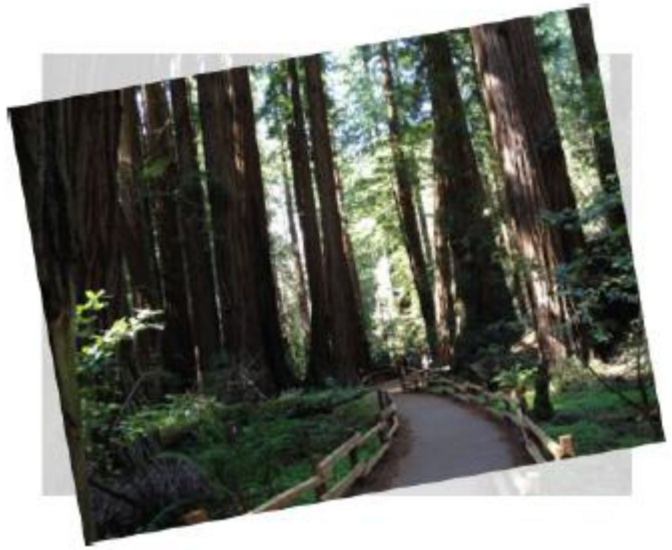
You can apply the **`transform`** property to the following element types:

- HTML elements with replaced content, such as **`img`**, **`canvas`**, form inputs, and embedded media
- Elements with **`display: block`**
- Elements with display: **`inline-block`**
- Elements with display: **`inline-table`** (or any of the **`table-*`** display types)

CSS Transforms : rotate

- Defines a 2D rotation, the angle is specified in the parameter
- **transform: rotate(angle)**
- Example

```
img {
width: 300px;
height: 400px;
transform: rotate(-10deg);
}
```



Rotating an **img** *element using* **transform: rotate().**

CSS Transforms : transform-origin

- Notice that the image rotates around its center point, which is the default point around which all transformations happen. But you can change that
- **transform-origin**

  Values: ***percentage*** **|** ***length*** **| left | center |**
  right | top | bottom

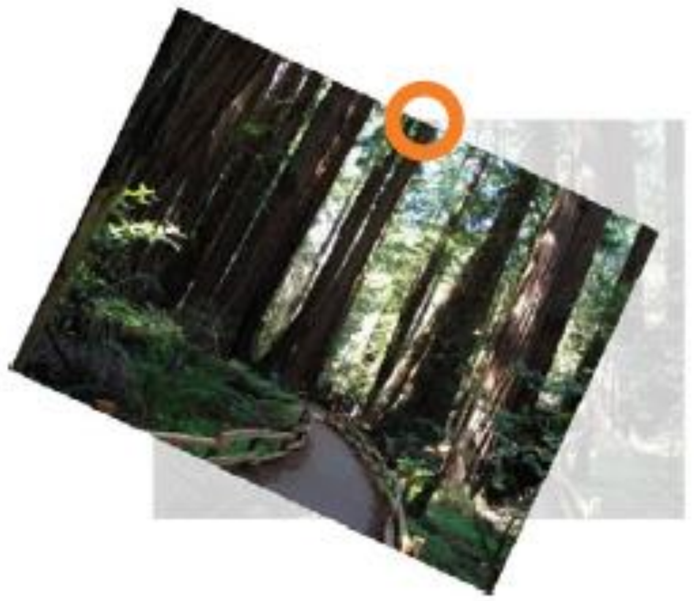
  Default: **50% 50%**
- Example

```
a img { transform-origin: center top;}
a img { transform-origin: 50%, 0%;}
a img { transform-origin: 150px, 0;}
```

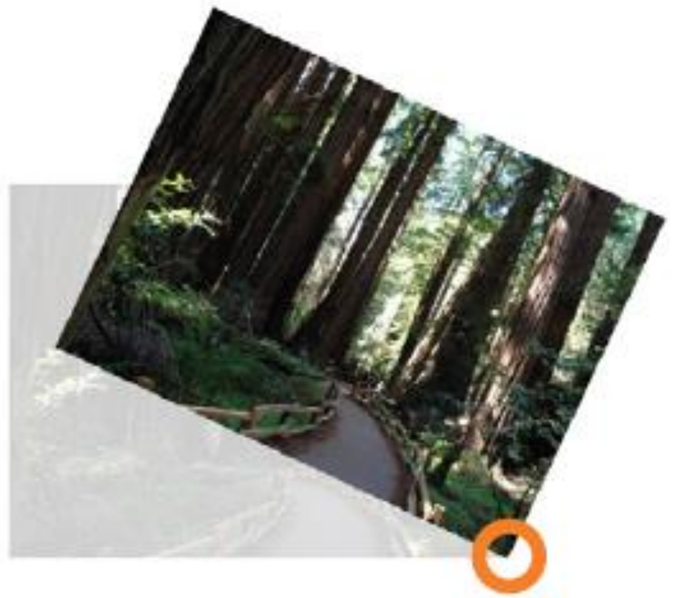
img *element rotated at the center point*

CSS Transforms : transform-origin

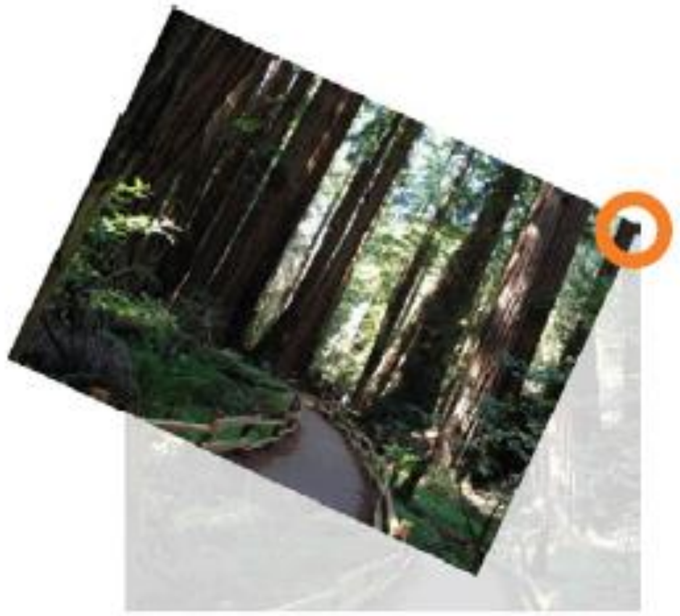
- The following images have all been rotated 25 degrees, but from different origin points



transform-origin: center top;



transform-origin: 100% 100%;



transform-origin: 400px 0

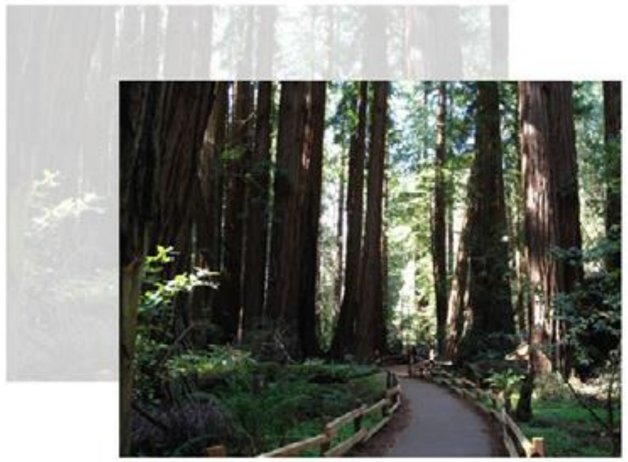
CSS Transforms : translate

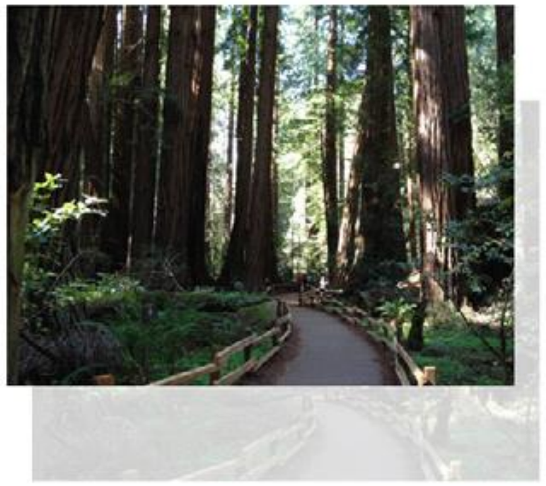
- moves an element from its current position (according to the parameters given for the X-axis and the Y-axis)
- **transform: translate(x,y)**

  transform: translateX(x);

  transform: translateY(y);
- Example

  a img { transform: translate(90px, 60px);} (1st image)
  a img { transform: translate(-5%, -25%); }(2nd image)





CSS Transforms : scale

- increases or decreases the size of an element (according to the parameters given for the width and height)
- **scale(x,y)**

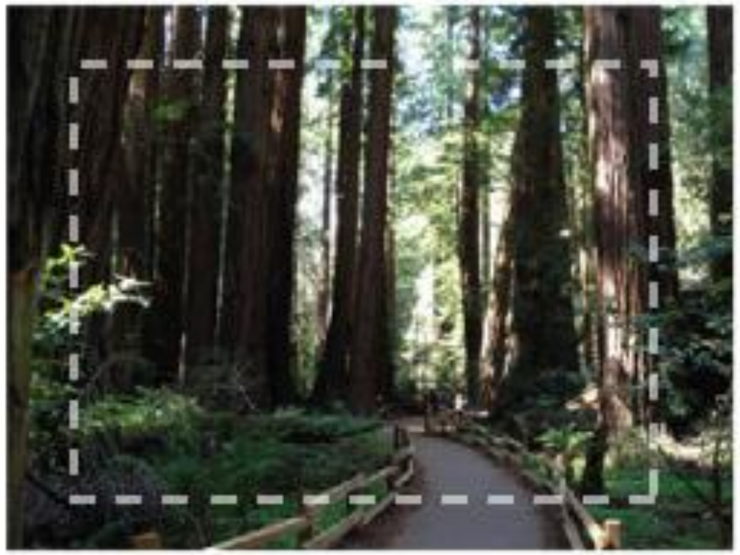
  scaleX(n)

  scaleY(n)
- Example

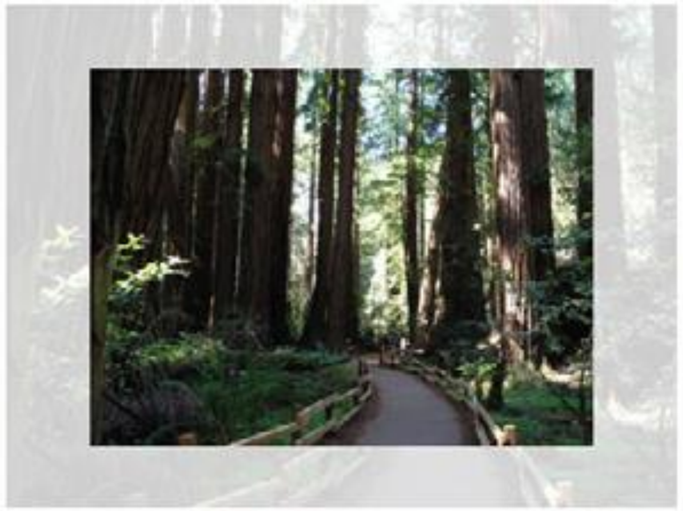
```
a img { transform: scale(1.25);}
a img { transform: scale(.75);}
a img { transform: scale(1.5, .5);}
```

CSS Transforms : scale

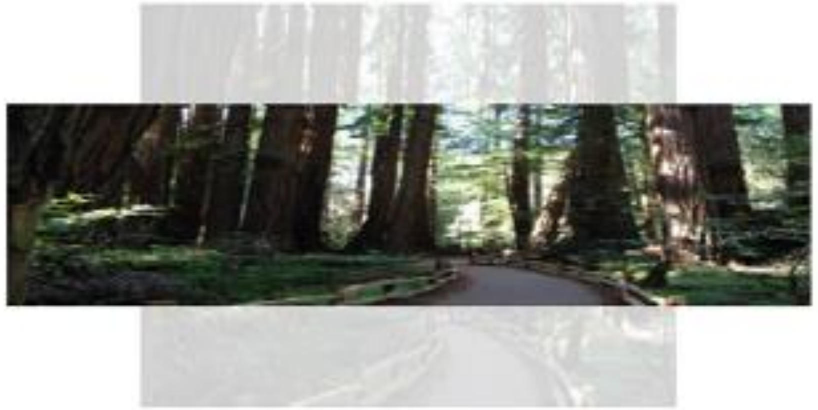
- Example of Scale function in transformation



transform: scale(1.25);



transform: scale(.75);



transform: scale(1.5, .5);

CSS Transforms : skew

- skews an element along the X and/or Y-axis by the given angles
- **skew(x-angle,y-angle)**

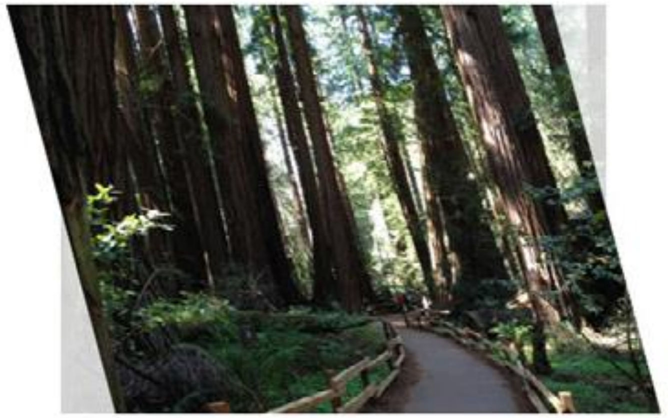
  skewX(angle)

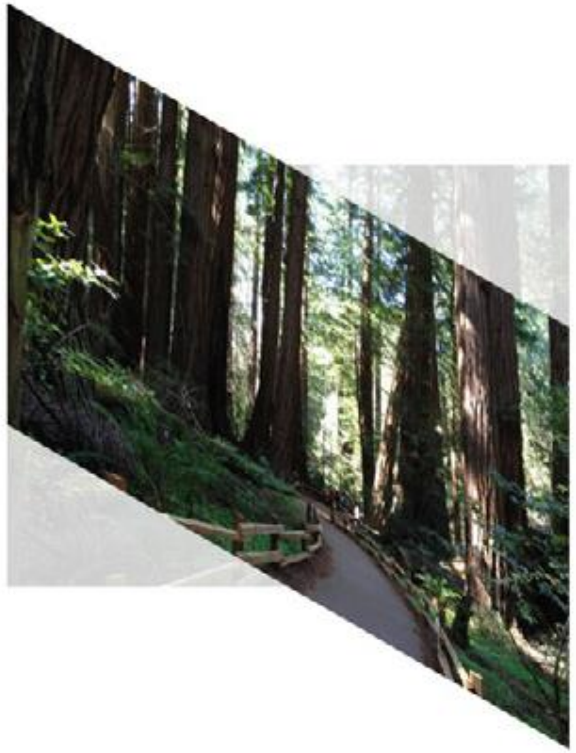
  skewY(angle)
- Example

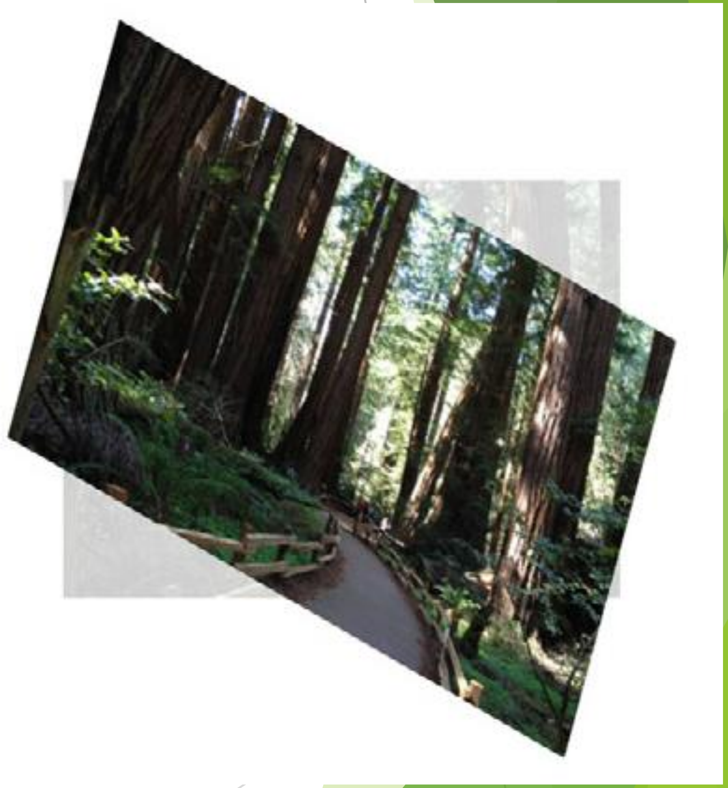
```
a img {transform: skewX(15deg);}
a img {transform: skewY(30deg);}
a img {transform: skew(15deg, 30deg);}
```

CSS Transforms : skew

- Example of Skew function in transformation







transform: skewX(15deg);

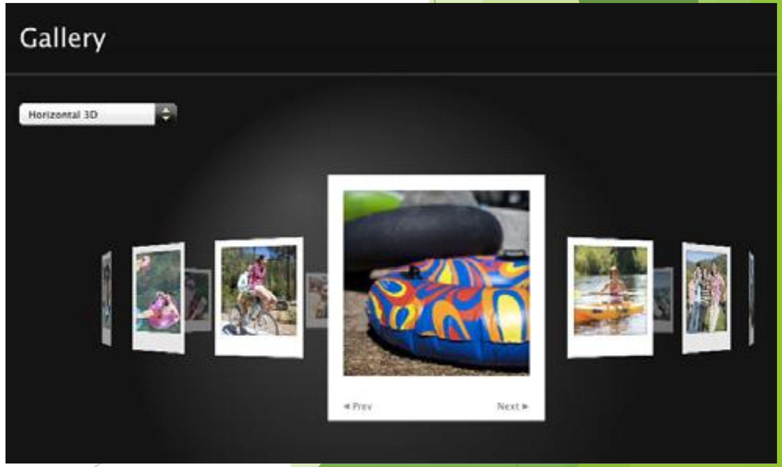
transform: skewY(30deg);

transform: skew(15deg, 30deg);

CSS 3D Transforms

- In addition, the CSS Transforms spec also describes a system for creating a sense of space and perspective.
- Combined with transitions, you can use 3-D transforms to create rich interactive interfaces
- such as image carousels, flippable cards, or *spinning cubes*
- *Demo: 3d spinning cube!!*

Learn how to make a cube
I design and build websites in Brighton



Exercise

- CSS3 transition, transformation, animation

References

- https://www.w3schools.com/