# Day 6

Transitions & Animations

Transitions in CSS are applied to an element and specify that when a property changes it should do so over gradually over over a period of time.

# CSS Transitions

You must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

```
a {
  background-color: orange;
}
a:hover { background-color: blue; }
```

```
a {
  background-color: orange;
  transition-property: background;
  transition-duration: 1s;
}
a:hover { background-color: blue; }
```

# CSS Transitions

*Transition declarations* *are* ***ALWAYS*** applied to the initial state.

This allows us to work with any other triggerings:

- :hover
- :focus
- :active
- :target
- :checked
- :disabled

# CSS Transitions

In order to reach the widest possible browser audience, designers often include a series of vendor prefix declarations as well as the generic declaration, all with the same values.

```
-webkit-transition-property: color;
-moz-transition-property: color;
-o-transition-property: color;
-ms-transition-property: color;
transition-property: color;
```

**Do NOT only include the -webkit- prefix**

# CSS Transition Syntax

```
transition:  [ <transition-property> ||
               <transition-duration> ||
               <transition-timing-function> ||
               <transition-delay> ]
```

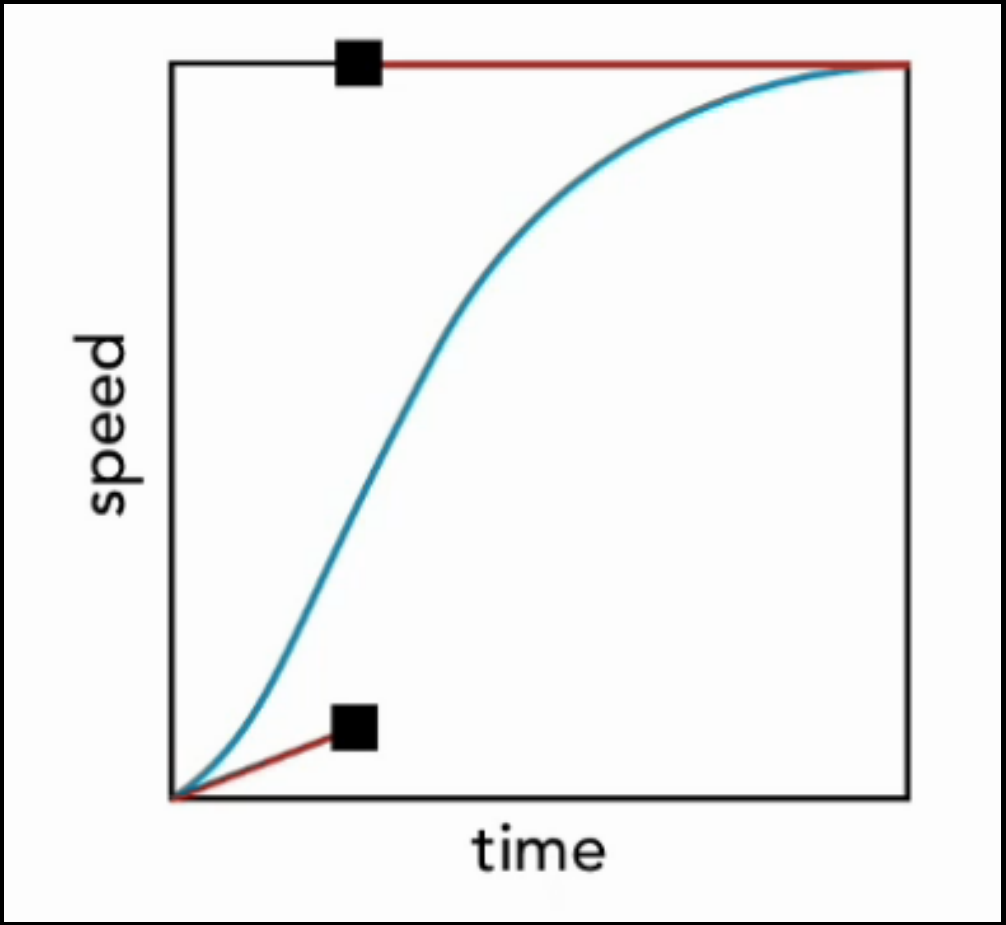
NO JAVASCRIPT
NEEDED!!!

# What can we transition?

| Property Name | Type |
| --- | --- |
| `background-color` | color |
| `background-image` | only gradients |
| `background-position` | percentage, length |
| `border-bottom-color` | color |
| `border-bottom-width` | length |
| `border-color` | color |
| `border-left-color` | color |
| `border-left-width` | length |
| `border-right-color` | color |
| `border-right-width` | length |
| `border-spacing` | length |

# CSS Transitions

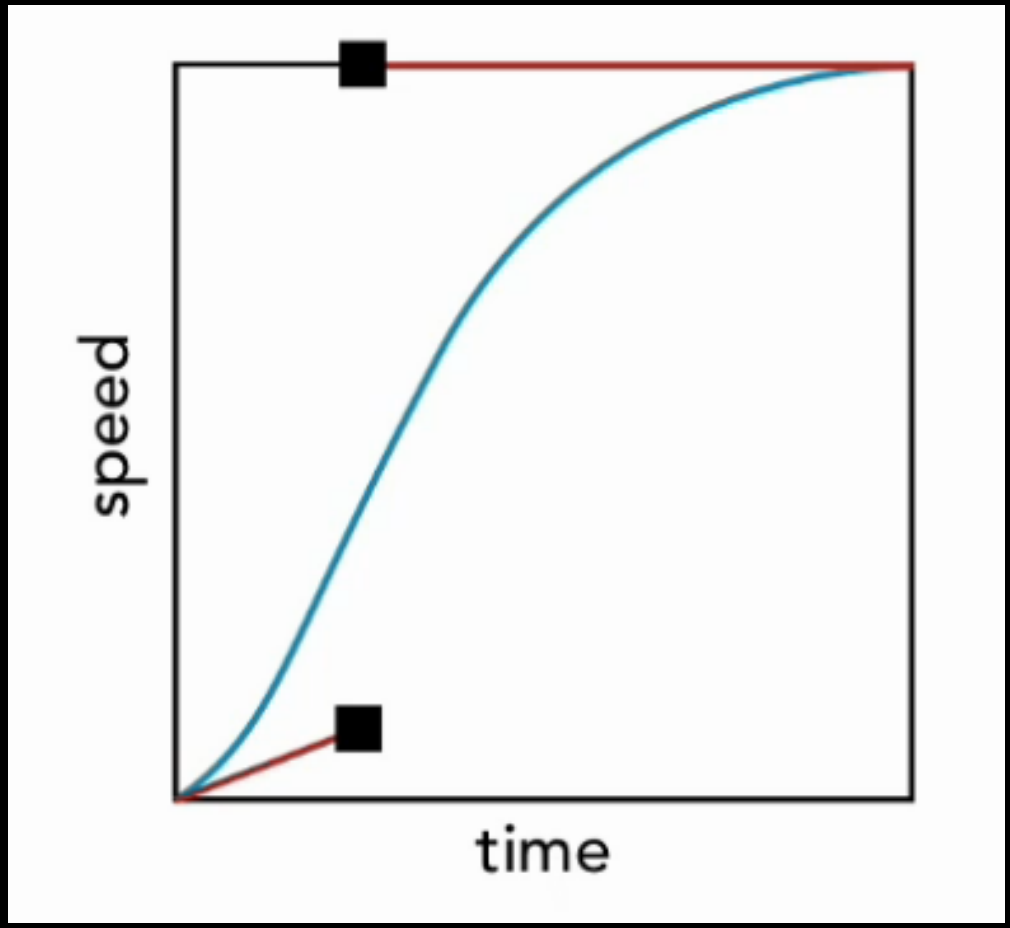
- default transitions start slow, speed up, and then slow down again
- Known as an “ease” transition



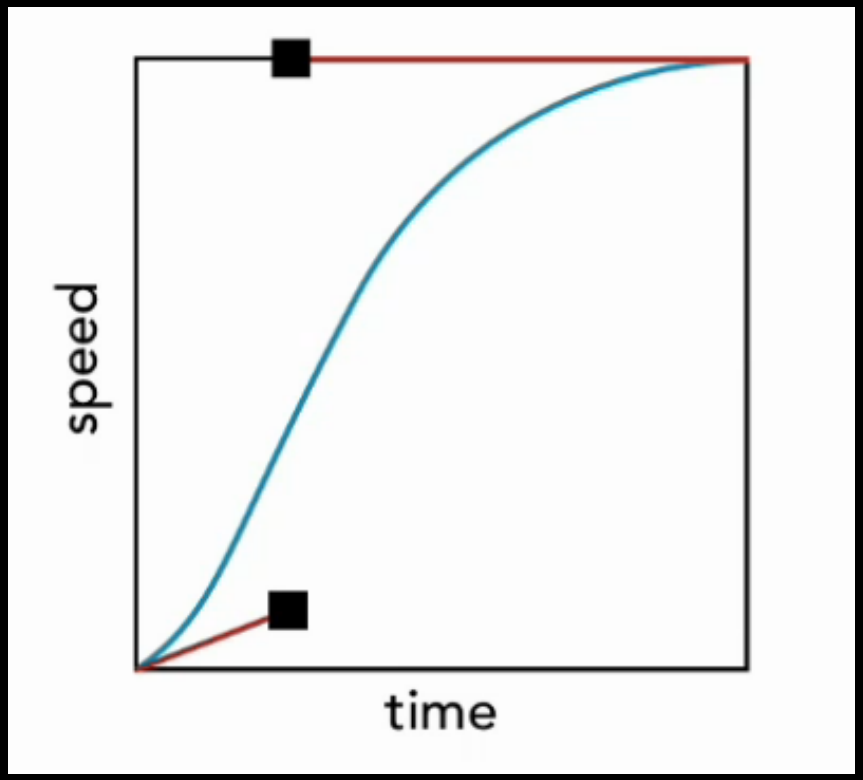
# transition-timing-function

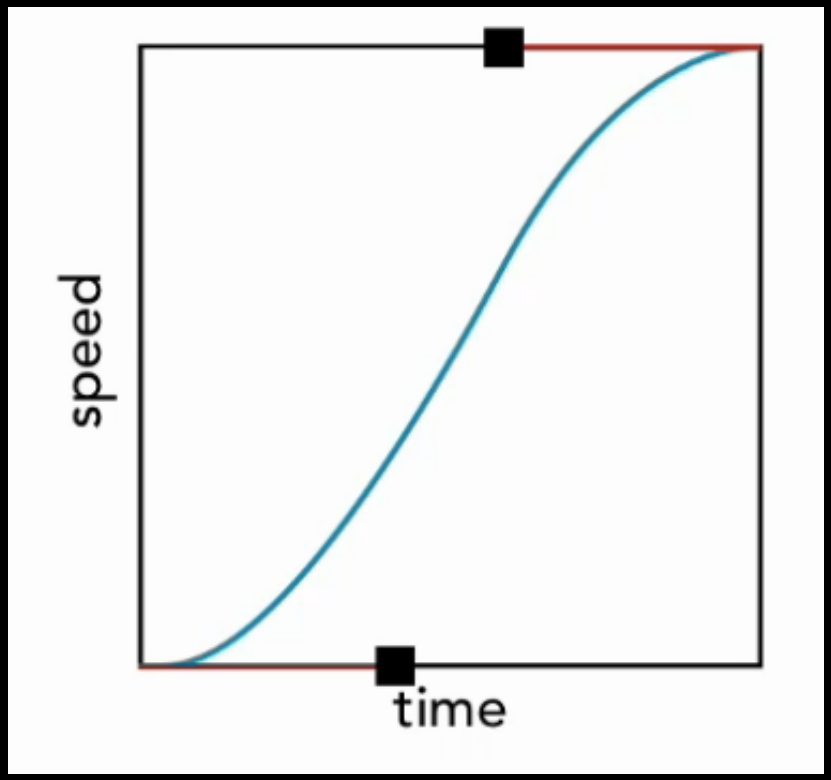
Values:

- ease (default)
- linear
- ease-in
- ease-out
- ease-in-out
- cubic-bezier



# transition-timing-function

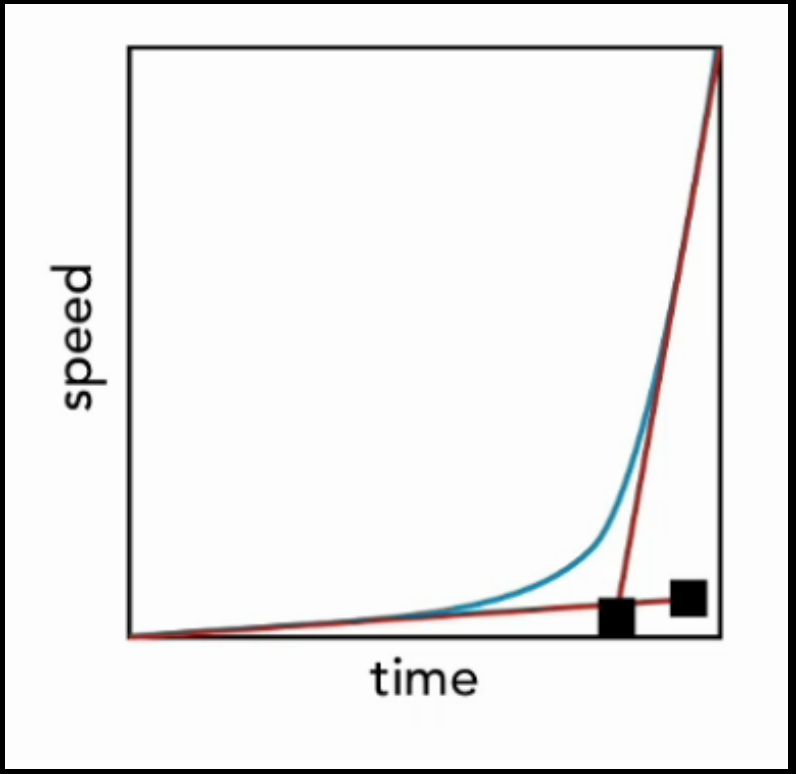


# transition-timing-function

cubic-bezier(x1,y1,x2, y2)

cubic-bezier(0.950, 0.050, 0.795, 0.035)



# transition-delay:

This let's you trigger things after an event has occurred.

Both transition delay and transition-timing function, you can use S for seconds or M-S for milliseconds.

```
transition-delay: 0.6s;
```

# Advanced transition-delay:

```
#dd_main2 {
  transition: all 1s ease-in-out;
}
```

```
#dd_main2 {
  transition-property: top, left;
  transition-duration: 1s, 1s;
  transition-delay: 0s, 1s;
}
```

What out come would these give us?

```
#dd_main2 {
  transition-property: top, left, border-radius, background-color;
  transition-duration: 2s, 1s, 0.5s, 0.5s;
  transition-delay: 0s, 0.5s, 1s, 1.5s;
}
```

# 2D Transforms

# Anatomy of A Transform

```
transform: function(parameters);
```

Four main functions:
Translate, Scale, Rotate, Skew

# transform: translate

```
transform: translate(10px,10px);
  transform: translateX(20px);
  transform: translateY(20px);
```

# transform: scale

```
transform: scale(.5,.5);

  transform: scaleX(2);

  transform: scaleY(.75);
```

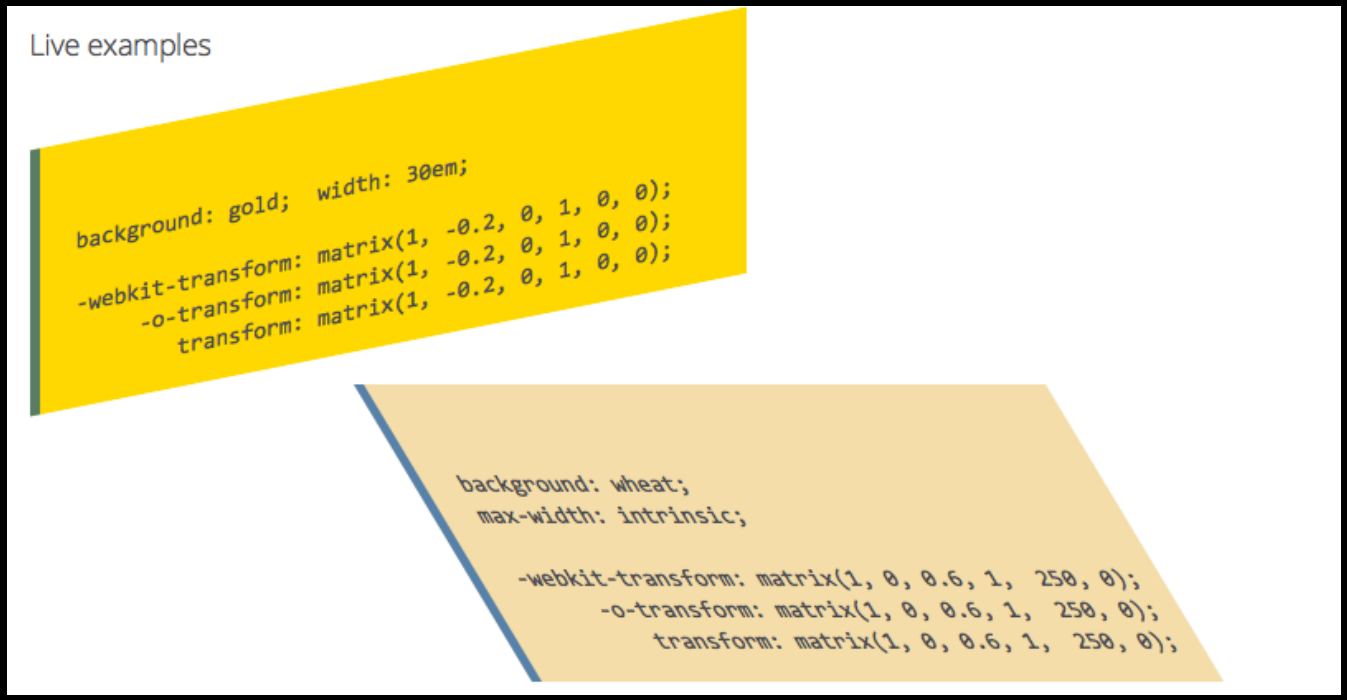
# transform: rotate

```
transform: rotate(15deg);
```

# transform: skew

```
transform: skewX(-25deg);

transform: skewY(10deg);
```



# transform-origin

```
transform-origin: 50% 50%; /* Center (Default) */

transform-origin: 10px 0; /* 10px right offset */

transform-origin: left bottom; /* Bottom left */
```

Lets you modify the origin for transformations of an element. For example, the transform-origin of the rotate() function is the center of rotation. (This property is applied by first translating the element by the negated value of the property, then applying the element's transform, then translating by the property value.)

# 3D transforms

# perspective:

```
perspective: 500px; /* mid-range from 200 to 1000 */

perspective: 1500px; /* very far away */

perspective: 150px; /* very close */
```

Perspective is essentially the distance from the viewer to the object. Lower numbers bring you closer to the object and distorted more. If you're a photographer, you can think of it as changing the focal length.

# perspective-origin:

```
perspective-origin: 50% 50%; /* center (default) */

perspective-origin: 25% 50%; /* mid-upper left */

perspective-origin: 50% 100%; /* center-bottom */
```

Determines the position the viewer is looking at. It is used as the *vanishing point* by the perspective property.

# transform: translate3d

```
transform: translate3d(20px,20px,-10px);

  transform: translateX(20px);

  transform: translateY(20px);

  transform: translateZ(-10px);
```

# transform: scale3d

```
transform: translate3d(20px,20px,-10px);

  transform: translateX(20px);

  transform: translateY(20px);

  transform: translateZ(-10px);
```

# transform: rotate3d

```
transform: rotate3d(10,20,20,15deg);

  transform: rotateX(10deg);

  transform: rotateY(20deg);

  transform: rotateZ(10deg);
```

# transform-style

```
transform-style: preserve-3d; /* default */

transform-style: flat; /* maps child-element on parent */
```

The transform-style CSS property determines if the children of the element are positioned in the 3D-space or are flattened in the plane of the element.

*If flattened, the children will not exist on their own in the 3D-space.*

Animations are different than transitions. When applied, they just run and do their thing. They offer more fine-grained control as you can control different stops of the animations.

NO JAVASCRIPT
NEEDED!!!

# CSS Animation

To use CSS animation, you first specify some keyframes for the animation - basically what styles will the element have at certain times. The browser does the tweening for you.

# CSS Animation

To use CSS animation, you first specify some keyframes for the animation - basically what styles will the element have at certain times. The browser does the tweening for you.

# CSS Animation Sub-properties

**`animation-delay`**
Configures the delay between the time the element is loaded and the beginning of the animation sequence.

**`animation-direction`**
Configures whether or not 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.

**`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.

**`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.

# In Class Example

```
h1 {
  animation-duration: 3s;
  animation-name: slidein;
}

@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
}
```