CSS Transitions, Positions & Animations - Complete Notes

1. CSS Transitions

CSS transitions allow you to change property values smoothly over a given duration.

Transition Properties

1. transition-property

Specifies which CSS properties should be transitioned.

```
transition-property: width;
transition-property: height;
transition-property: scale;
transition-property: background;
transition-property: all; /* transitions all animatable properties */
```

2. transition-duration

Specifies how long the transition should take.

```
transition-duration: 1s;
transition-duration: 1000ms;
transition-duration: 0.5s;
```

3. transition-timing-function

Specifies the speed curve of the transition effect.

```
transition-timing-function: ease;        /* slow start + fast + slow end */
transition-timing-function: ease-in;     /* slow start + linear */
transition-timing-function: ease-out;    /* linear + slow end */
transition-timing-function: ease-in-out; /* slow start + linear + slow end */
transition-timing-function: linear;      /* same speed throughout */
```

4. transition-delay

Specifies when the transition effect will start.

```
transition-delay: 0s;    /* no delay */
transition-delay: 1s;    /* 1 second delay */
transition-delay: 1000ms; /* 1000 milliseconds delay */
```

Shorthand Syntax

```
transition: property duration timing-function delay;
/* Examples */
transition: scale 0.5s ease-in-out 0s;
transition: width 2s linear 0.5s;
transition: all 1s ease-in-out;
```

Time Conversion Reference

- **1 hour = 60 minutes**
- **1 minute = 60 seconds**
- **1 hour = 3600 seconds** ★ Important
- **1 second = 1000 milliseconds** ★ Important
- **3600 seconds = 3,600,000 milliseconds**

2. CSS Position Property

The position property specifies the type of positioning method used for an element.

Position Values

1. static (default)

- Default positioning
- Elements flow normally in the document
- top, right, bottom, left, z-index have no effect

```
position: static;
```

2. relative

- Positioned relative to its normal position
- Creates a new positioning context for child elements
- Can use top, right, bottom, left, z-index

```
position: relative;
top: 10px;
left: 20px;
```

3. absolute

- Positioned relative to the nearest positioned ancestor
- Removed from normal document flow
- If no positioned ancestor, positioned relative to the initial containing block
- Can use top, right, bottom, left, z-index

```
position: absolute;
top: 0;
right: 0;
```

4. fixed

- Positioned relative to the viewport
- Stays in the same place even when scrolled
- Removed from normal document flow
- Can use top, right, bottom, left, z-index

```
position: fixed;
top: 0;
left: 0;
```

5. sticky

- Toggles between relative and fixed positioning
- Positioned relative until it crosses a specified threshold
- Then it becomes fixed
- Can use top, right, bottom, left, z-index

```
position: sticky;
top: 0;
```

Positioning Properties

Available for relative, absolute, fixed, and sticky positions:

- **top**: Distance from the top edge
- **right**: Distance from the right edge
- **bottom**: Distance from the bottom edge
- **left**: Distance from the left edge
- **z-index**: Stacking order (higher values appear on top)

Parent-Child Relationships

- **relative** and **absolute**: Parent-child relationship
- **fixed** and **sticky**: Positioned relative to viewport/scroll

---

3. CSS Animations

CSS animations allow you to animate CSS properties over time with more control than transitions.

Animation Properties

1. animation-name

Specifies the name of the @keyframes rule.

```
animation-name: slideIn;
animation-name: fadeOut;
```

2. animation-duration

Specifies how long the animation should take.

```
animation-duration: 2s;
animation-duration: 1500ms;
```

3. animation-timing-function

Specifies the speed curve of the animation.

```
animation-timing-function: ease;
animation-timing-function: ease-in;
animation-timing-function: ease-out;
animation-timing-function: ease-in-out;
animation-timing-function: linear;
animation-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);
```

4. animation-delay

Specifies when the animation will start.

```
animation-delay: 0s;
animation-delay: 1s;
animation-delay: 500ms;
```

5. animation-iteration-count

Specifies how many times the animation should run.

```
animation-iteration-count: 1;        /* run once */
animation-iteration-count: 3;        /* run 3 times */
animation-iteration-count: infinite; /* run forever */
```

6. animation-direction

Specifies whether the animation should play forward, backward, or alternate.

```
animation-direction: normal;    /* forward */
animation-direction: reverse;   /* backward */
animation-direction: alternate; /* forward then backward */
animation-direction: alternate-reverse; /* backward then forward */
```

7. animation-fill-mode

Specifies what styles apply before/after the animation.

```
animation-fill-mode: none;      /* default */
animation-fill-mode: forwards;  /* keep final keyframe styles */
animation-fill-mode: backwards; /* apply first keyframe styles during delay
*/
animation-fill-mode: both;      /* apply both forwards and backwards */
```

8. animation-play-state

Specifies whether the animation is running or paused.

```
animation-play-state: running; /* default */
animation-play-state: paused;
```

Shorthand Syntax

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

/* Examples */
animation: slideIn 2s ease-in-out 0s 1 normal forwards running;
animation: fadeOut 1s linear infinite;
animation: bounce 0.5s ease-in-out 0s infinite alternate;
```

@keyframes Rule

Defines the animation sequence by specifying CSS styles at various points.

```
@keyframes slideIn {
  0% {
    transform: translateX(-100%);
    opacity: 0;
  }
  50% {
    transform: translateX(-10%);
    opacity: 0.5;
  }
  100% {
    transform: translateX(0);
    opacity: 1;
  }
}
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
```

Complete Example

```css
.animated-box {
  width: 100px;
  height: 100px;
  background: blue;
  position: relative;

  /* Animation */
  animation: moveAndFade 3s ease-in-out 0s infinite alternate;

  /* Transition for hover effects */
  transition: background-color 0.3s ease;
}

.animated-box:hover {
  background-color: red;
}

@keyframes moveAndFade {
  0% {
    left: 0;
    opacity: 0.3;
    transform: scale(1);
  }
  50% {
    left: 200px;
    opacity: 1;
    transform: scale(1.2);
  }
  100% {
    left: 400px;
    opacity: 0.3;
    transform: scale(1);
  }
}
```

Key Differences

Transitions vs Animations

- **Transitions**: Triggered by state changes (hover, focus, etc.), simple A to B movement
- **Animations**: Can run automatically, complex sequences with multiple keyframes

Position Context

- **static**: No positioning context
- **relative**: Creates positioning context, element remains in document flow
- **absolute**: Uses nearest positioned ancestor as reference, removed from flow
- **fixed**: Uses viewport as reference, removed from flow
- **sticky**: Hybrid of relative and fixed based on scroll position

---

Browser Support Notes

- All modern browsers support these properties
- Use vendor prefixes for older browsers if needed:
  - `-webkit-` for Chrome/Safari
  - `-moz-` for Firefox
  - `-ms-` for Internet Explorer

Performance Tips

- Use `transform` and `opacity` for better performance
- Avoid animating layout properties (width, height, margin, padding)
- Use `will-change` property to optimize animations
- Consider using CSS animations over JavaScript for simple effects