PER SCHOLAS

Introduction to Transitions and Media Query

PER SCHOLAS

Learning Objectives

This lesson provides an introduction to CSS transitions and media query. By the end of this lesson, learners will be able to:

- Apply transitions to an HTML page.
- Control page rendering via the viewport meta tag.
- Use the @media CSS at-rule to customize CSS for varying and dynamic scenarios.

PER SCHOLAS

Table of Contents

PER SCHOLAS

View in Slideshow Mode

This presentation contains animations, and is best viewed in slideshow mode.

Slideshow ▼

or Ctrl + F5

CSS Transitions

CSS transitions provide a way to create gradual transitions between the values of CSS properties. These transitions can be controlled by specifying their timing function, duration, and other attributes.

For example, if you change the color of an element from white to black, usually the change is instantaneous. With CSS transitions enabled, changes occur over a period of time, which can be customized. In the example case, we would transition from white, through shades of grey, and eventually to black.

You can also animate things like size, position, and a number of other properties. For a full list of animatable CSS properties, click here.

Fun with CSS Transitions!

PER SCHOLAS

CSS Transition Properties

To create transitions, you will need to make use of the following properties:

- `transition`: A shorthand property that encompases all of the below properties.
- `transition-delay`: Specifies the duration to wait before starting a property's transition effect when its value changes.
- `transition-duration`: Sets the length of time a transition animation should take to complete.
- `transition-property`: Sets which CSS properties a transition effect will be applied to.
- `transition-timing-function`: Sets how intermediate values are calculated for properties being affected by the transition effect.

PER SCHOLAS

Transition Example: Background Color

In this example, the background color will change from green to lightskyblue over a duration of two seconds. Using the **`:hover`** pseudo-class, we can specify that the element should change when the user hovers their mouse over it.

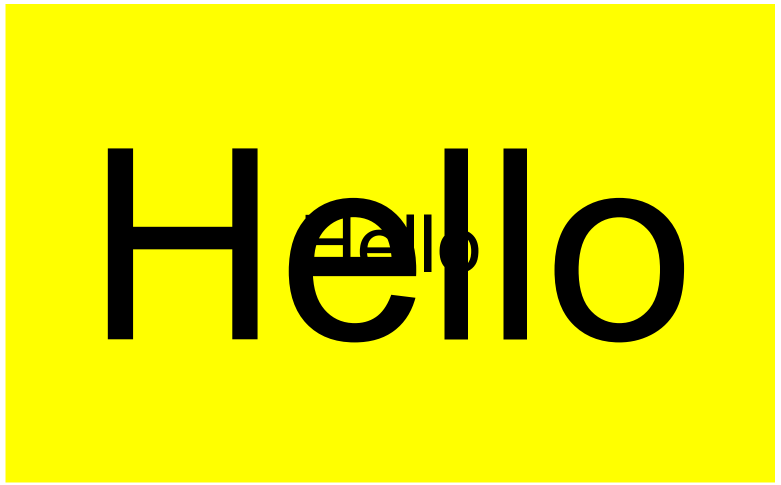
```css
#box1 {
   background-color: green;
   transition-property: background-color;
   transition-duration: 2s;
}
#box1:hover {
   background-color: lightskyblue;
}
```

PER SCHOLAS

Transition Example: Font

In this example, the font size will change from 30px to 100px over a duration of two seconds. Using the **`:hover`** pseudo-class, we can specify that the element should change when the user hovers their mouse over it.

```
#box2 {
   background-color: yellow;
   transition-property: font-size;
   transition-duration: 2s;
}
#box2:hover {
   font-size: 100px;
}
```

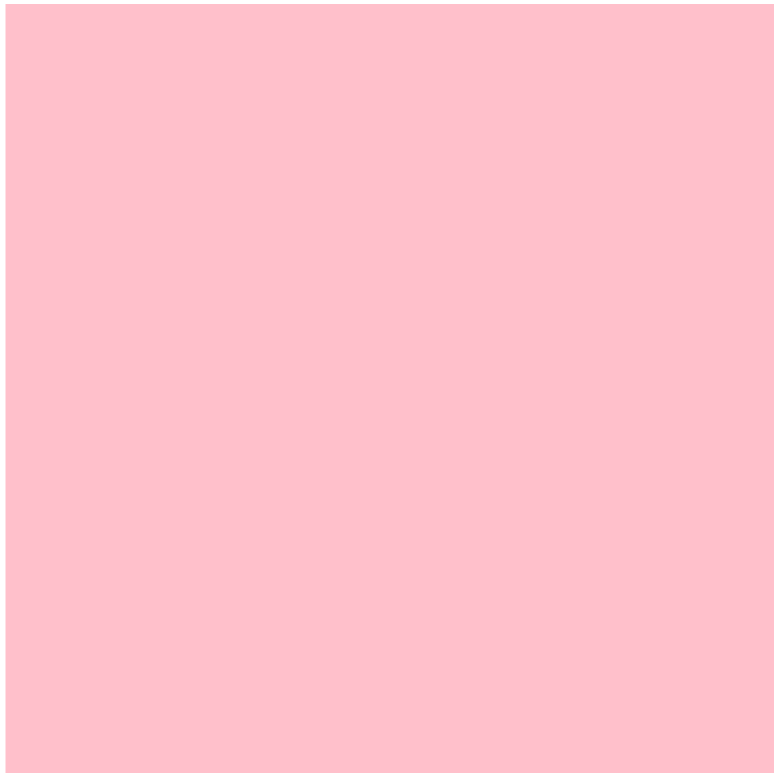


PER SCHOLAS

Transition Example: Size

In this example, the width and height will be changed from 100px to 200px over the course of two seconds. Using the **`:hover`** pseudo-class, we can specify that the element should change when the user hovers their mouse over it.

```css
#box3 {
   background-color: pink;
   transition-property: width, height;
   transition-duration: 2s;
}
#box3:hover {
   width: 200px;
   height: 200px;
}
```

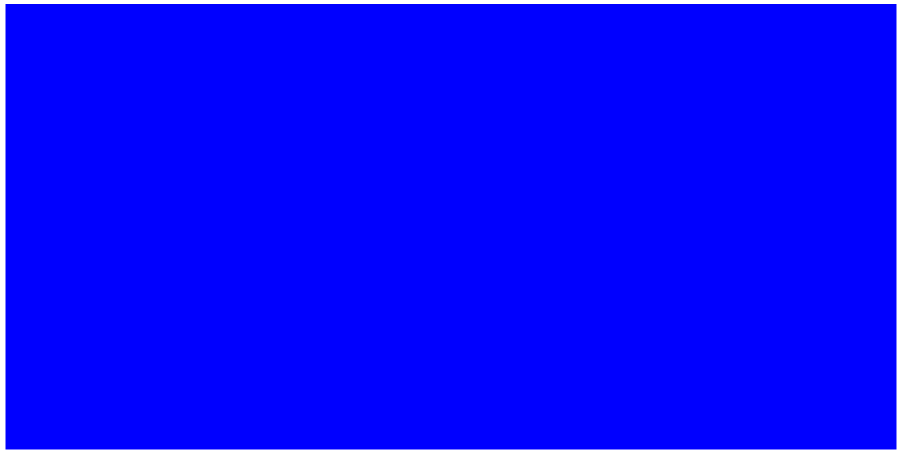


PER SCHOLAS

Transition Example: Position

In this example, the box will translate 100px to the right over the course of 0.6 seconds. The `transform` CSS property lets you rotate, scale, skew, or translate an element, and the `translate()` CSS function repositions an element horizontally or vertically.

```
#box4 {
   background-color: blue;
   transition-duration: 0.6s;
}
#box4:hover {
   transform: translateX(100px);
}
```



PER SCHOLAS

Transition Example: Position Two

In this example, the `rotate()` CSS function will rotate the box 360 degrees over the course of three seconds.

```
#box5 {
   background-color: red;
   transition-duration: 3s;
}
#box5:hover {
   transform:rotate(360deg);
}
```

PER SCHOLAS

Transition Example: Multiple Properties

You can add multiple properties to the transition-property property by separating them with a comma. In this example, the box will change its font color from black to white, its background color from white to black, its border color from blue to orange, and rotate 360 degrees, all over the course of one second.

```
#box6 {
   color: black;
   background-color: white;
   border-color: #4284F4;
   transition-property: color,
background-color, border-color;
   transition-duration: 1s;
}
#box6:hover {
   color: white;
   background-color: black;
   border-color: #FFAB40;
   transform:rotate(360deg);
}
```

Fun with CSS Transitions!

Transition Shorthand

The `transition` property is a shorthand property encompassing all four transition properties. The syntax for this shorthand property is: **`transition: <property> <duration> <timing-function> <delay>;`**.

- ➢ The first parameter specifies to which property the transition will apply.
- ➢ The second parameter specifies how long the transition will last.
- ➢ The third parameter specifies a built-in CSS timing function. This function will affect the way the property will change. Some of the functions are: **`ease`**, **`ease-in`**, **`ease-in-out`**, **`ease-out`**, **`linear`**, **`step-end`**, **`step-start`**, and **`steps()`**.
- ➢ The fourth parameter specifies when the transition starts after the property has changed.
- ➢ Just like other properties, you do not have to specify every parameter. Parameters that are not specified will use their default values. The most commonly declared parameters are the first (property) and second (duration).

PER SCHOLAS

Responsive Design Principles

Responsive Design is the practice of making sure your content looks good on all screen sizes. Everything in the website, including layouts, fonts, and images, should automatically adapt to the user's device.

In the early 2000s, developers focused on making sure their websites looked good on larger screen sizes like laptops and desktop computers. In today's world, you have to consider smaller devices like mobile phones, tablets, and even watches.

When building content with responsive design in mind:

- ➢ Do not set large fixed sizes (width) to elements; this may cause horizontal scrolling on small devices. Try to use a responsive value such as percentage.
- ➢ Use CSS media queries to apply different styles for different-sized screens.

Media queries are an important component of responsive design, which we will go over in detail during this lesson.

PER SCHOLAS

Viewport

A viewport is the representation of the area in computer graphics that is currently being viewed by the user. In web browser terms, it refers to the part of the document that is currently visible in its window (or the screen, if the document is being viewed in full screen mode). Content outside of the viewport is not visible on screen until scrolled into view.

The viewport meta tag allows you to control the size and scale of the viewport, telling the browser how the page should be rendered. For example:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Pages optimized for a variety of devices must include a meta viewport tag in the head of the document. A meta viewport tag gives the browser instructions on how to control the page's dimensions and scaling:

- The **`width=device-width`** tells the browser to match the width of the device visiting the site (device-independent pixels).
- The **`initial-scale=1`** tells the browser to set a 1:1 relationship between CSS pixels and device-independent pixels. This is due to multiple devices having different screen sizes and resolutions. It also disables user scaling. Other options for scale include: **`minimum-scale`**, **`maximum-scale`**, and **`user-scalable`**.

PER SCHOLAS

Media Query: Overview

Media queries are simple filters that can be applied to CSS styles. Styles declared inside of a media query will only be applied when that media query is met. This conditional application of CSS properties allows for powerful, adaptive, responsive design.

Media types used in media queries include:

- **`all:`** Suitable for all devices.
- **`print:`** Intended for paged material and documents viewed on a screen in print-preview mode.
- **`screen:`** Intended primarily for screens.
- **`speech:`** Intended for speech synthesizers.

The basic syntax for writing a media query in CSS is:

```css
@media media-type (media-feature) {
   /* Styles go here. */
}
```

Source: Wikimedia Commons

PER SCHOLAS

Media Query: Syntax

```
@media media-type (media-feature) {
   /* Styles go here. */
}
```

Here we will look at this syntax in more detail:

- **`@media`** is a type of *at-rule* in CSS. These rules dictate what the CSS will look like based on conditions.
- **`media-type`** refers to the category of media for the device. The different media types include **`all`**, **`print`**, **`screen`** and **`speech`**, as shown on the previous slide. **`media-type`** is optional, and the **`all`** type is implied.
- **`media-feature`** section can include a wide variety of values for querying things like **`max-width`**, **`orientation`**, and much more. The most commonly queried features are:
  - **`min-width`:** Rules applied for any browser width greater than the value defined in the query.
  - **`max-width`:** Rules applied for any browser width less than the value defined in the query.
  - **`min-height`:** Rules applied for any browser height greater than the value defined in the query.
  - **`max-height`:** Rules applied for any browser height less than the value defined in the query.
  - **`orientation=portrait`:** Rules applied for any browser where the height is greater than or equal to the width.
  - **`orientation=landscape`:** Rules for any browser where the width is greater than the height.

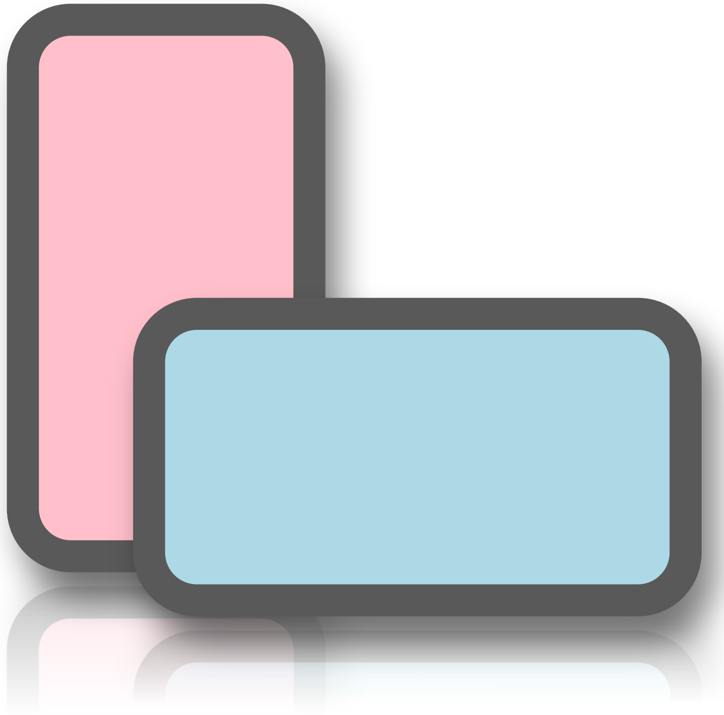
PER SCHOLAS

Media Query Example: Orientation

The example below shows the background color being changed depending on the current orientation of the user's screen. In landscape orientation the background color will be light blue, while in portrait orientation it will be pink.

```
@media only screen and (orientation: landscape) {
  body {
    background-color: lightblue;
  }
}

@media only screen and (orientation: portrait) {
  body {
    background-color: pink;
  }
}
```



PER SCHOLAS

CSS Breakpoints and Target Device Sizes

Should you write separate media queries for *every single device* on the market?

The short answer to that question is no.

There are way too many devices out on the market to try to write a media query for each device. Technology is always changing, which means new devices will always be coming out with new properties and sizes.

It is more important that you **target a range of devices**.

Here is list of some common breakpoints used for media queries:

- **320px-480px:** Mobile devices
- **481px-768px:** iPads, Tablets
- **769px-1024px:** Small screens, laptops
- **1025px-1200px:** Desktops, large screens
- **1201px and larger:** Extra large screens, TV

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones,
600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up)
*/
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops,
1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

PER SCHOLAS

Media Query: Best Practices

When using media queries, abide by these best practices:

- ➢ Use media queries to apply styles based on device characteristics.
- ➢ Use **`min-width`** over **`min-device-width`** to ensure the broadest experience.
- ➢ Use relative sizes for elements to avoid breaking layout.
- ➢ Do not let the screen size determine what content you should hide; let the content make that determination.
- ➢ Use the **`media`** attribute in the **`<link>`** tag with one of the media types.
  - ○ `<link rel="stylesheet" href="main.css" media="print">`
- ➢ Create CSS stylesheets based on major breakpoints.
- ➢ Take care of minor breakpoints as well, so that pages look more natural.
- ➢ Use relative units such as percentages, instead of finite units like pixels.
- ➢ Keep line text to a maximum of 70 to 80 characters.
- ➢ Design for mobile devices first, and then expand.

PER SCHOLAS

Responsive Design Examples

Responsive Image Grid:

➔ Download the source code for the responsive image grid example.

In this example, we use media queries together with flexbox to create a responsive image gallery.

Responsive Website:

➔ Download the source code for the responsive website example.

In this example, we use media queries together with flexbox to create a responsive website with a navigation bar and flexible content.

Knowledge Check

- What is a CSS transition and how is it used?
- How is transition-property used?
- What are two CSS properties that are animatable?
- What values does the transition shorthand property take?
- What is responsive design?
- What is a viewpoint?
- How do you start a media query?
- What are the four media types used in media queries?
- What are two orientations for a mobile device?

PER SCHOLAS

Summary

CSS transitions are powerful tools that can add movement and dynamism to your webpages. You can transition many different CSS properties, and transition multiple properties at once. The `transition` shorthand property allows you to quickly write transitions in one line of CSS.

Responsive design is a core concept and component of good web design, and requires the content to adapt to whatever device it is being delivered to:

- ➢ The viewport is where content is displayed to the user, and understanding how content layout is controlled within the viewport is a key component of responsive web design.
- ➢ Media queries can be used to write conditional CSS in order to create responsive layouts across a variety of device types and sizes.
- ➢ You should not attempt to account for every device on the market when designing web pages; instead, use common breakpoints to limit yourself to a few different layouts.

PER SCHOLAS

Questions?