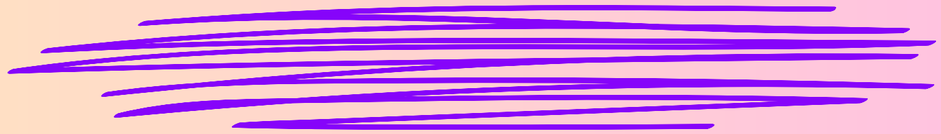
Cascading and Specificity

Cascading and Specificity in CSS

CSS uses a system called "Cascading" to determine which styles apply to elements when there are multiple rules. The three main concepts here are Inheritance, Cascading, and Specificity.



Inheritance

Some CSS properties are naturally inherited by child elements, such as color, font-family, and text-align.

- Other properties like padding, margin, and width are *not* inherited by default.

html

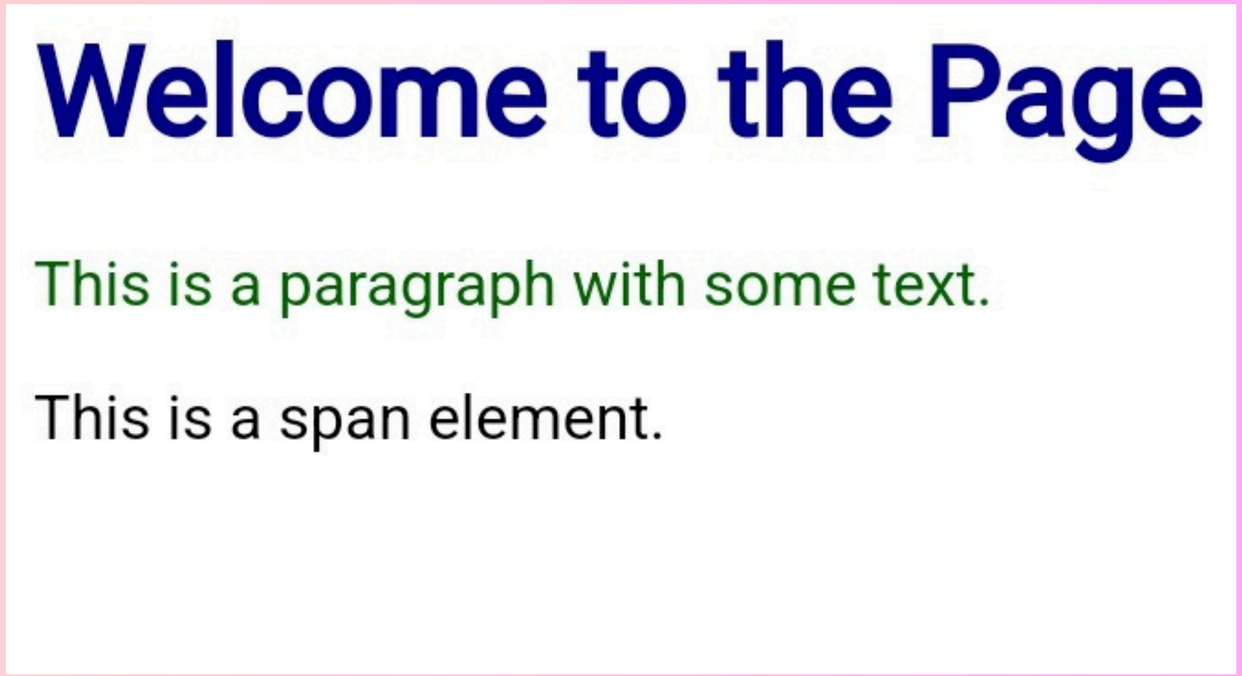
```html
<div class="content">
  <h1>Welcome to the Page</h1>
  <p>This is a paragraph with some text.</p>
  <span>This is a span element.</span>
</div>
```

css

```css
.content {
  font-family: Arial, sans-serif;
  font-size: 18px;
}

h1 {
  color: darkblue;
}

p {
  color: darkgreen;
}
```



Control Values

- Inherit: Forces an element to inherit the value from its parent, even if it wouldn't naturally.

html Copy code

```html
<div class="parent">
  <p>This is a paragraph.</p>
  <span>This is a span that should inherit the font color.</span>
</div>
```

css Copy code

```css
.parent {
  color: blue; /* Color set on the parent */
}

span {
  color: inherit; /* Forces span to inherit color from the parent */
}
```

Control Values

- Initial: Resets a property to its default value.

html Copy code

```html
<div class="container">
  <p>This is a paragraph with default styling.</p>
</div>
```

css Copy code

```css
.container {
  color: green; /* Color set on the container */
}

p {
  color: initial; /* Resets color to the default value (usually black) */
}
```

Control Values

- Unset: Resets a property to inherit if it is naturally inheritable or to its initial value if it is not.

html Copy code

```html
<div class="parent">
  <p>This is a paragraph with a set color.</p>
  <span>This span will use unset to determine its color.</span>
</div>
```

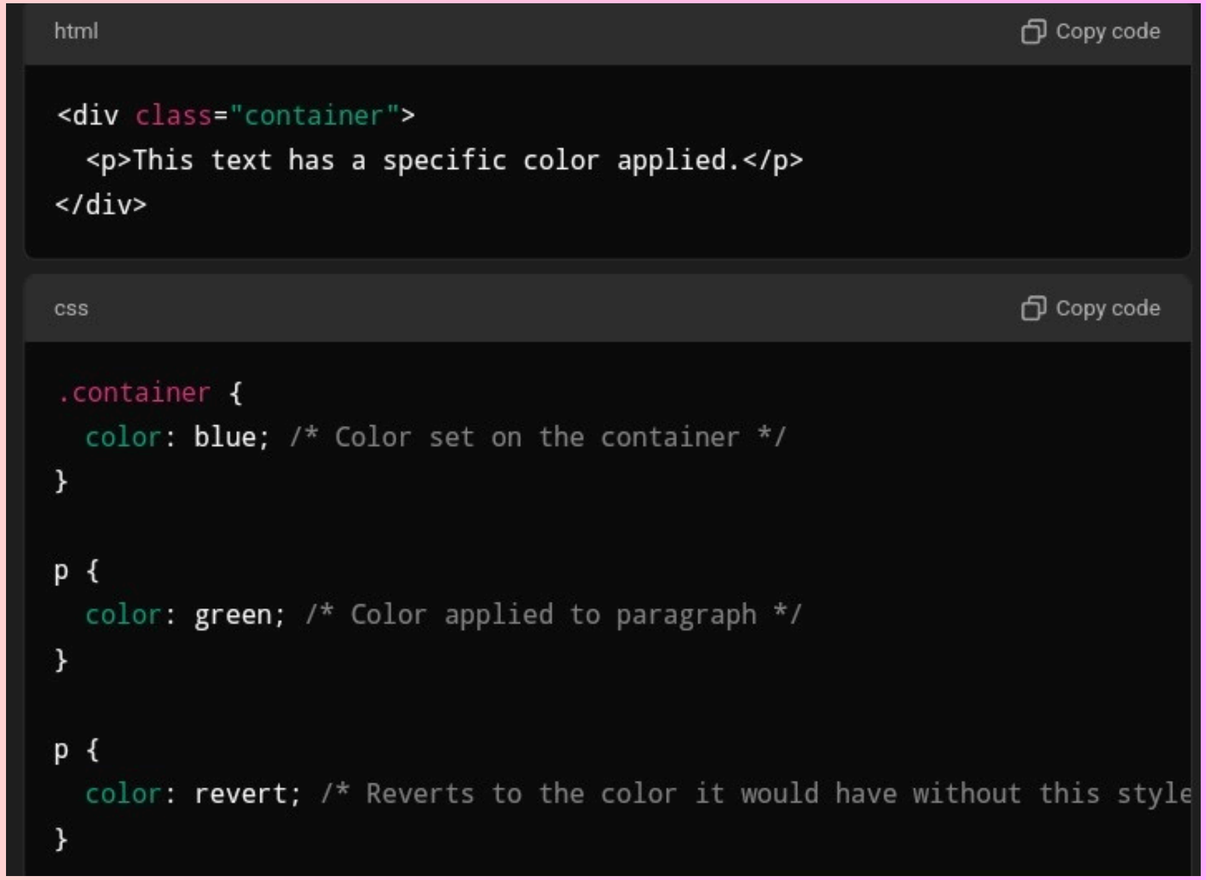
css Copy code

```css
.parent {
  color: red; /* Set color on the parent */
}

span {
  color: unset; /* Resets color based on inheritance or defaults */
}
```

Control Values

- Revert: Reverts a property to the value it would have from the user agent stylesheet or the CSS rules applied earlier.



html Copy code

```html
<div class="container">
  <p>This text has a specific color applied.</p>
</div>
```

css Copy code

```css
.container {
  color: blue; /* Color set on the container */
}

p {
  color: green; /* Color applied to paragraph */
}

p {
  color: revert; /* Reverts to the color it would have without this style
}
```

Cascading

Cascading determines which rule is applied if multiple CSS rules could apply to the same element.

- - CSS rules have *source order*:
- - Browser defaults (like user agent styles)
- - User-defined styles (styles you write)
- - Inline styles (style added directly within an HTML tag)
- - Styles marked as !important override all others.

```css
p {
  color: red;
}

p {
  color: blue;
}
```

Specificity

Specificity is a measure of how specific a CSS rule is, deciding which rule takes precedence.

- It follows a point-based system
- Inline styles (in the HTML tag) have the highest specificity.
- IDs (example) are more specific than classes (.example), attributes, or pseudo-classes (:hover).
- Classes, attributes, and pseudo-classes are more specific than elements (like div, h1)

html

Copy code

```html
<div id="main" class="content">
  <p>This paragraph will have the color determined by the most specific r
</div>
```

css

Copy code

```css
p {
  color: green;
}

#main p {
  color: blue;
}

.content p {
  color: red;
}
```

This paragraph will have the color determined by the most specific rule.

The specificity of selectors is calculated in the order of inline styles, then ID selectors ,class selectors, and finally element selectors. The higher the specificity, the more likely a rule will be applied

Importance (!important)

The !important flag overrides any other rule, regardless of specificity or source order. When used, this rule takes the highest

```css
p {
  color: red !important;
}

p {
  color: blue;
}
```

html

```html
<div id="main" class="content">
  <p>This text color will be determined by cascading rules.</p>
</div>
```

css

```css
p {
  color: black;
}

.content p {
  color: green;
}

#main p {
  color: blue;
}

#main p {
  color: red !important;
}
```

This text color will be determined by cascading rules.

To determine which CSS rule applies to an element, the browser follows this cascading order:

- Importance (!important): Rules with !important will override all others.
- Specificity: If no !important is present, the browser looks at specificity. More specific rules (like IDs) will override less specific ones (like classes or tags).
- Source Order: If specificity is the same, the last rule in the code is applied.

Chapter 18: Colors in CSS

CSS provides several ways to specify colors, which you can use to style text, backgrounds, borders, and more.

Here are the most common formats:

NAMED COLORS

CSS supports around 140 named colors like red, blue, green, black, white, etc.

HEXADECIMAL COLORS

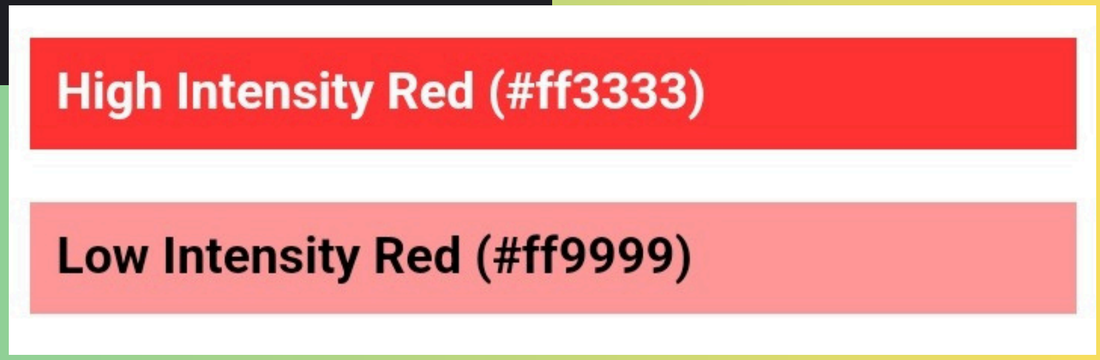
Hex codes are a popular format for colors in CSS. For example, #FF5733 represents a shade of orange and other shade has 6 digits.

INTENSITY

The brightness or dullness of a color.

INTENSITY

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Color Intensity Example</title>
</head>
<body>
<h2 style="background-color: #ff3333; color: #ffffff; padding: 10px;"
  >High Intensity Red (#ff3333)</h2>
<h2 style="background-color: #ff9999; color: #000000; padding: 10px;"
  >Low Intensity Red (#ff9999)</h2>
</body>
</html>
```



Here are the most common formats:

Named Colors
- CSS supports around 140 named colors like red, blue, green, black, white, etc.

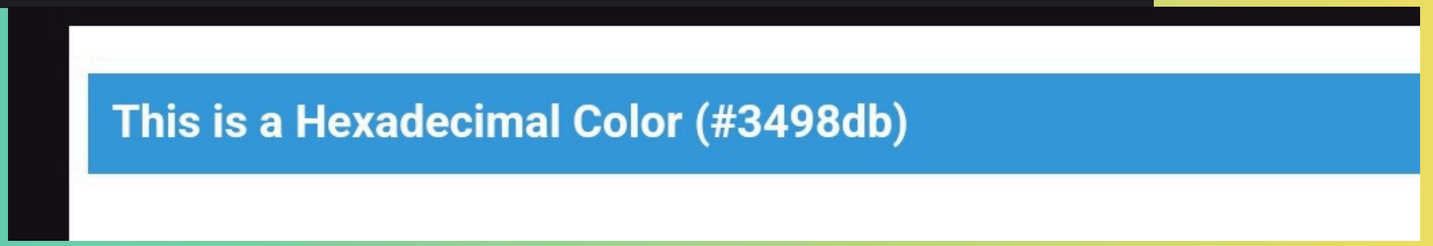
Hexadecimal Colors
- Hex codes are a popular format for colors in CSS. For example, #FF5733 represents a shade of orange every color has 6 digits.

HEXADECIMAL COLORS

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Hexadecimal Color Example</title>
</head>
<body>

<h2 style="background-color: #3498db; color: #ffffff; padding: 10px;"
  >This is a Hexadecimal Color (#3498db)</h2>

</body>
</html>
```



RGB and RGBA Colors

- RGB stands for Red, Green, and Blue and is written like this: rgb(255, 0, 0) for pure red.
- RGBA is similar, but the "A" stands for *Alpha* (opacity): rgba(255, 0, 0, 0.5) for a semi-transparent red.

HSL and HSLA Colors

- HSL stands for Hue, Saturation, and Lightness. For example, hsl(0, 100%, 50%) is pure red.
- HSLA adds an alpha channel for transparency, like hsla(0, 100%, 50%, 0.5) for a semi-transparent red.

Opacity

- opacity is a CSS property used to control the transparency of an element, ranging from 0 (fully transparent) to 1 (fully opaque).

RGB and RGBA Colors

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>RGB and RGBA Colors Example</title>
</head>
<body>
<h2 style="background-color: rgb(255, 0, 0); color: #ffffff; padding: 10px;"
  >RGB Color (rgb(255, 0, 0))</h2>
<h2 style="background-color: rgba(255, 0, 0, 0.5); color: #000000; padding: 10px;"
  >RGBA Color (rgba(255, 0, 0, 0.5))</h2>
</body>
</html>
```

HSL AND HSLA Colors & Opacity

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HSL, HSLA, and Opacity Examples</title>
    <style>
        /* 1. HSL Color */
        .hsl-color {
            background-color: hsl(200, 70%, 50%);
            color: white;
            padding: 20px;
            margin-bottom: 10px;
        }

    /* 2. HSLA Color */
    .hsla-color {
        background-color: hsla(150, 60%, 50%, 0.5);
      /* Green with 60% saturation, 50% lightness, 50% opacity */
        color: black;
        padding: 20px;
        margin-bottom: 10px;
    }

    /* 3. Opacity Example */
    .opacity-example {
        background-color: #ff6347; /* Tomato color */
```

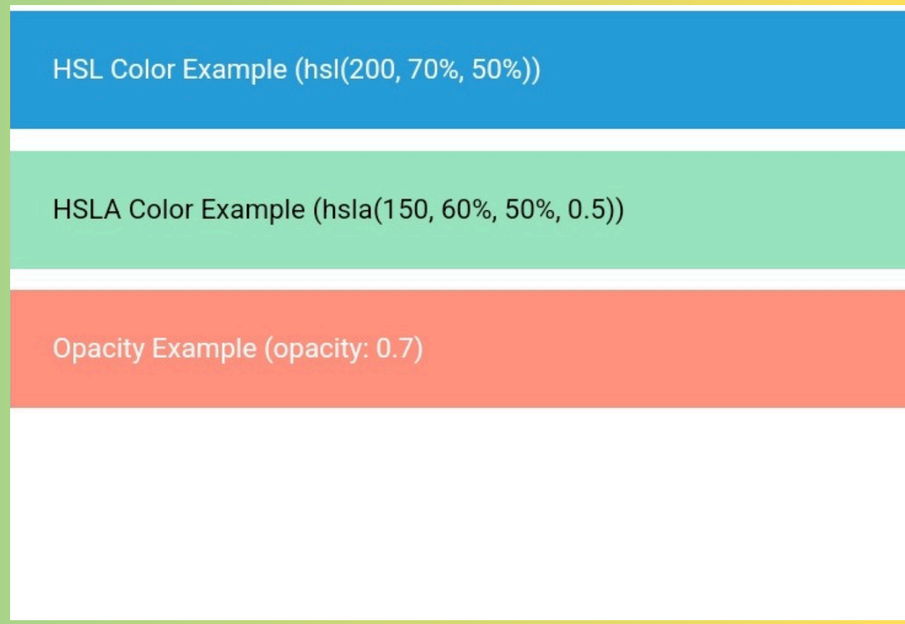
```
        /* 3. Opacity Example */
        .opacity-example {
            background-color: #ff6347; /* Tomato color */
            color: white;
            opacity: 0.7; /* 70% opacity */
            padding: 20px;
            margin-bottom: 10px;
        }
    </style>
</head>
<body>

    <div class="hsl-color">HSL Color Example (hsl(200, 70%, 50%))</div>
    <div class="hsla-color">HSLA Color Example (hsla(150, 60%, 50%, 0.5))</div>
```

```
            padding: 20px;
            margin-bottom: 10px;
        }
    </style>
</head>
<body>

    <div class="hsl-color">HSL Color Example (hsl(200, 70%, 50%))</div>
    <div class="hsla-color">HSLA Color Example (hsla(150, 60%, 50%, 0.5))</div>
    <div class="opacity-example">Opacity Example (opacity: 0.7)</div>

</body>
</html>
```



ANOTHER EXAMPLE.

EXAMPLE 1

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
      color: #333; /* Base color */
    }

    .text {
      color: rgb(255, 0, 0); /* Override color with RGB */
    }

    #important {
      color: blue !important; /* Highest specificity due to !important */
    }
  </style>
</head>
<body>
  <p class="text">This text is red because of the class selector.</p>
  <p id="important">This text is blue due to the use of !important.</p>
</body>
</html>
```

This text is red because of the class selector.

This text is blue due to the use of !important.

EXAMPLE 2

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Cascading and Specificity Example</title>
    <style>
        /* General element selector */
        p {
            color: blue; /* Applies to all <p> elements initially */
        }

        /* Class selector */
        .text {
            color: green; /* More specific than the element selector */
        }

        /* ID selector */
        #main-title {
            color: red; /* ID selectors have higher specificity */
        }

        /* Inline style (highest specificity) */
        h1 {
            color: purple; /* Would be overridden by #main-title */
        }
    </style>
</head>
<body>
    <div class="container">
        <h1 id="main-title">Hello World</h1>
        <p class="text">This is a paragraph with a class.</p>
        <p>This is another paragraph without a class.</p>
    </div>
</body>
</html>
```

Hello World

This is a paragraph with a class.

This is another paragraph without a class.

This example shows how CSS decides which colors to apply based on rule strength. Here, the !important rule on the #important ID makes the text blue, ignoring other color rules.

HANDS ON:

This is a regular paragraph inside the container. It should follow the container styles.

This is an important paragraph inside the container. It should be bold and dark green due to specificity.

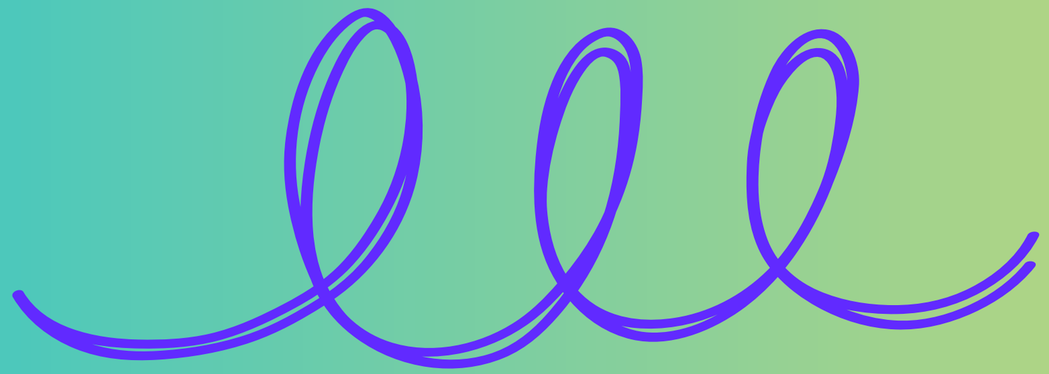
This is a highlighted and important paragraph inside the container. With ID specificity, it should be orange, italic, underlined, and bold.

This is a paragraph with the highest priority styles, due to !important. It should appear red and larger, despite other rules.

This is a regular paragraph outside the container, following only the default paragraph styles.

This is an important paragraph outside the container, which should appear in dark blue and bold.

JOHN REIL REBAYLA & EUGENE SERVITO



Thanks.