LECTURE 5

CSS Fundamentals

The language that controls how HTML looks: colours, fonts, spacing, and layout.

CSC336 Web Technologies

OVERVIEW

In This Lecture

1. CSS syntax, and the three ways to attach CSS to HTML
2. Selectors: element, class, id, attribute, grouping, descendant
3. Pseudo-classes and pseudo-elements
4. The cascade and specificity — every conflict case, and the real scoring method
5. Which properties inherit from parent to child, and which do not
6. The most common colour, font, text, and background properties

What Is CSS?

1. A style sheet language: HTML says what something IS, CSS says how it should look
2. Separates content (HTML) from presentation (CSS)
3. Change a whole site's look by editing one CSS file
4. The same HTML can be restyled for print, mobile, or accessibility

SYNTAX

A CSS Rule

```
p {
  color: darkslategray;
  font-size: 16px;
}
```

A selector (which elements) plus a declaration block (what style) — every declaration is a property: value pair, ending in a semicolon.

1. Inline Styles

```
<p style="color: red; font-weight: bold;">This paragraph is styled inline.</p>
```

This paragraph is styled inline.

2. Internal Style Sheets

```
<head>
  <style>
    body { font-family: Arial, sans-serif; }
    h1 { color: navy; }
  </style>
</head>
<body>
  <h1>Welcome</h1>
</body>
```

2. Internal Style Sheets, Rendered

A <style> block inside <head> — applies to the whole page, but stuck in that one HTML file.

3. External Style Sheets — Preferred

TIP

One file, many pages: every page links the same styles.css, so the whole site looks consistent and you edit one file to restyle it. Separation of concerns keeps HTML focused on content. Browsers cache the file, so pages after the first load faster. Designers and developers can work on CSS and HTML independently.

Class Selector: .highlight

```
<p class="highlight">This text is important.</p>

.highlight {
  background-color: yellow;
}
```

ID Selector: #main-header

```
<div id="main-header">Site Title</div>

#main-header {
  font-size: 2rem;
  text-align: center;
}
```

Class vs. ID

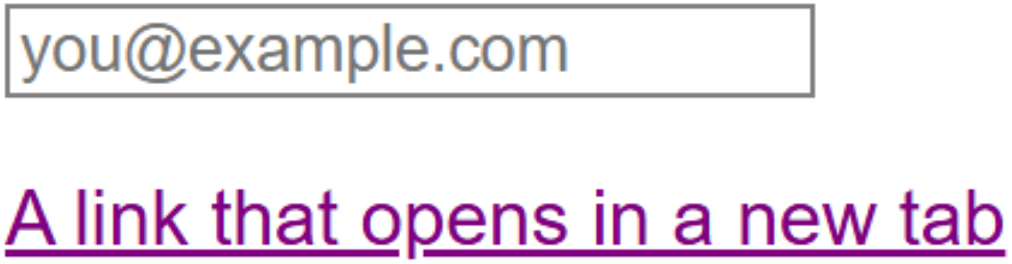
WARNING

Use a class when a style might apply to more than one element (most of the time). Use an id only for something that truly appears once on the page. Overusing ids makes CSS harder to reuse.

Attribute Selector

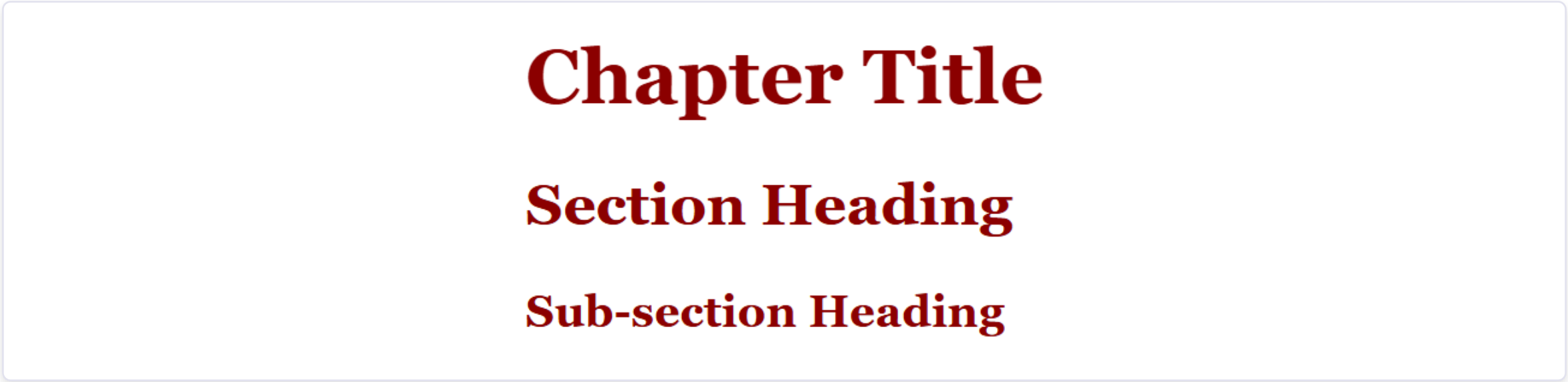
```
input[type="email"] {
  border: 1px solid gray;
}

a[target] {
  color: purple;
}
```



Grouping Selector

```
h1, h2, h3 {
  font-family: Georgia, serif;
  color: darkred;
}
```



Descendant Selector

```
article p {
  color: #333;
}

<article>
  <p>This paragraph IS styled (inside article).</p>
</article>
<p>This paragraph is NOT styled (outside article).</p>
```

This paragraph IS styled (it's inside article).

This paragraph is NOT styled (it's outside article).

The Universal Selector

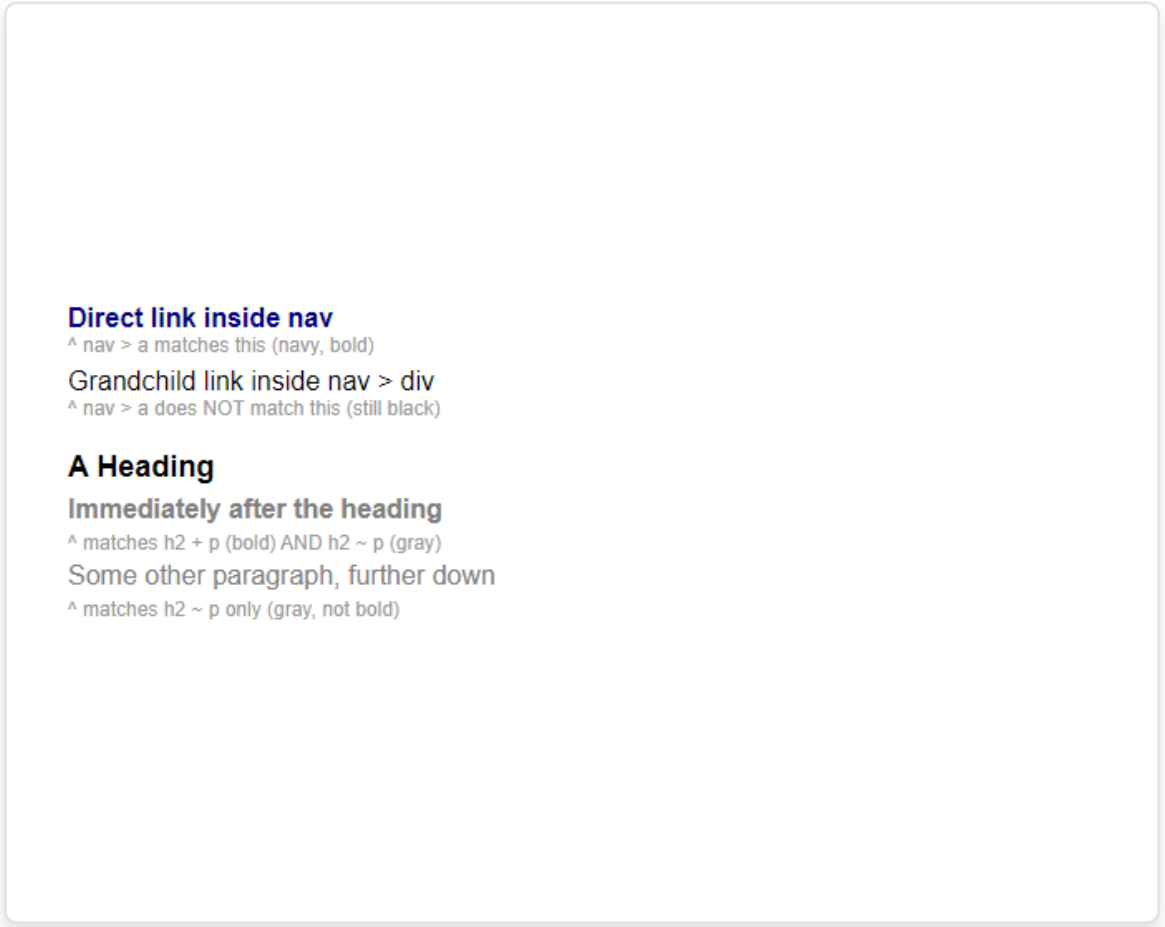
```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

* matches every single element on the page — most commonly seen in a "CSS reset" at the very top of a stylesheet.

Combinators: Child and Sibling

```
nav > a { color: navy; font-weight: bold; }
h2 + p  { font-weight: bold; }
h2 ~ p  { color: gray; }

<nav>
  <a>Direct link</a>
  <div><a>Grandchild link</a></div>
</nav>
<h2>A Heading</h2>
<p>Immediately after</p>
<p>Further down</p>
```



Common Pseudo-Classes

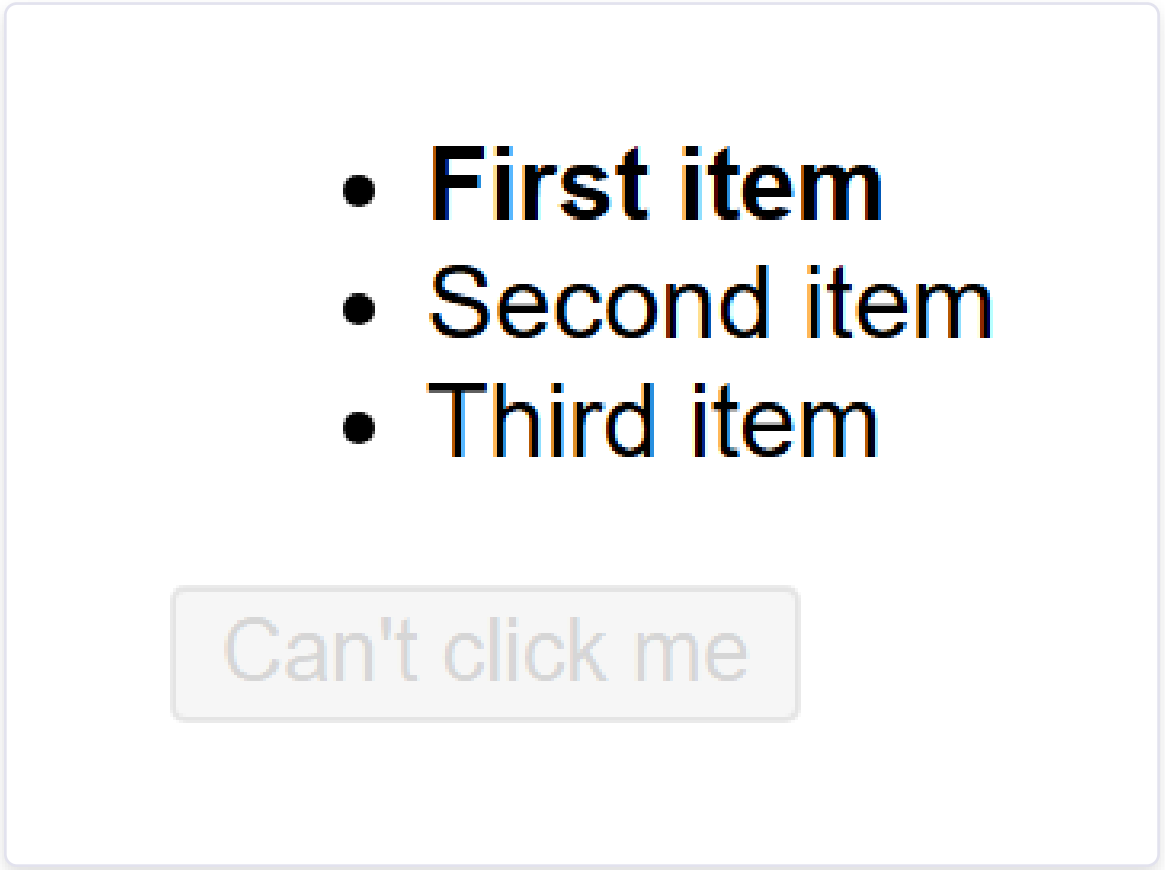
Pseudo-class	Matches
:hover / :focus	Mouse over the element / the element is focused
:active	While the element is being clicked
:first-child / :last-child	First / last child of its parent
:nth-child(n)	The nth child of its parent
:not(selector)	Elements that do NOT match the given selector

Pseudo-Classes, Rendered

li:first-child and button:disabled are states the browser can render up front.

```
li:first-child {
  font-weight: bold;
}

button:disabled {
  opacity: 0.5;
}
```



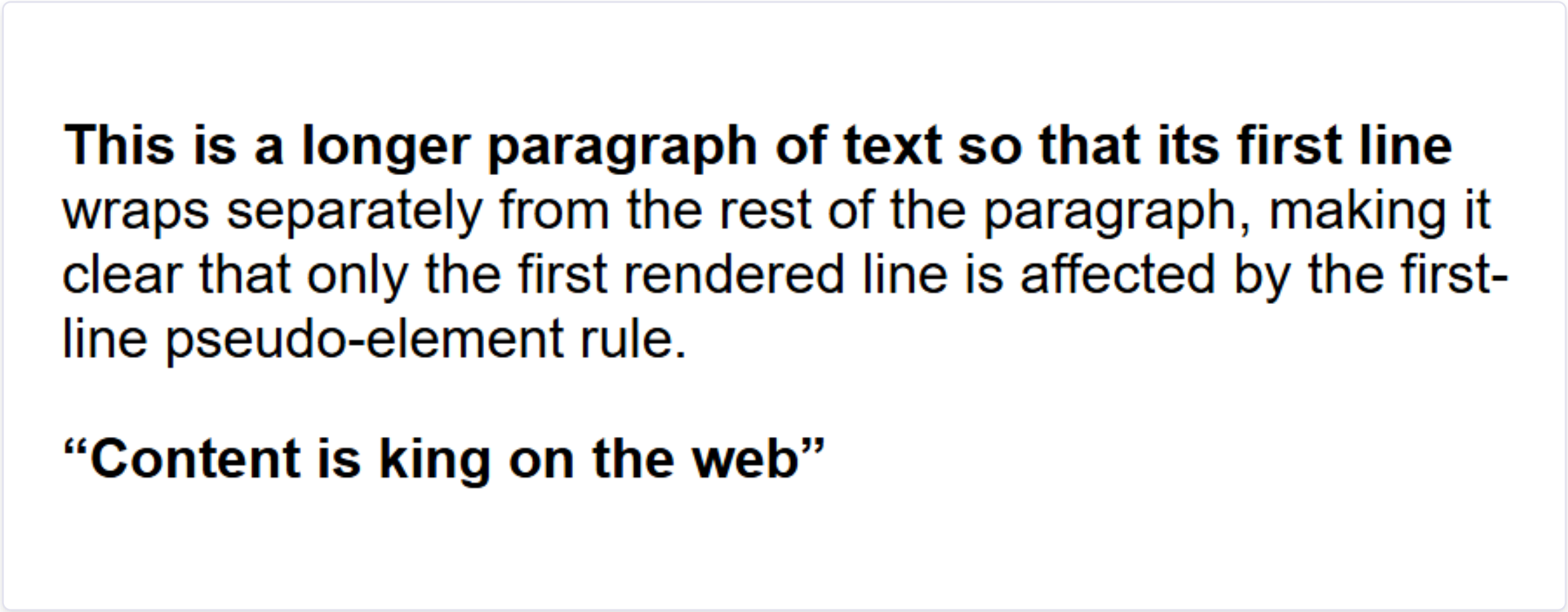
Pseudo-Elements

```
p::first-line { font-weight: bold; }

.quote::before { content: "\201C"; }
.quote::after  { content: "\201D"; }
```

::before and ::after insert content that is not in the HTML at all — purely for decoration.

Pseudo-Elements, Rendered



Specificity, From Lowest to Highest

Selector Type	Example
Element / pseudo-element	p, ::before
Class / attribute / pseudo-class	.highlight, [type="text"], :hover
ID	#main-header
Inline style	style="..."
!important	Overrides everything else

Specificity Decides the Winner

```
p { color: black; }              /* low specificity */
.intro { color: green; }         /* medium          */
#lead-paragraph { color: blue; } /* high            */

<p id="lead-paragraph" class="intro">What colour am I?</p>
```

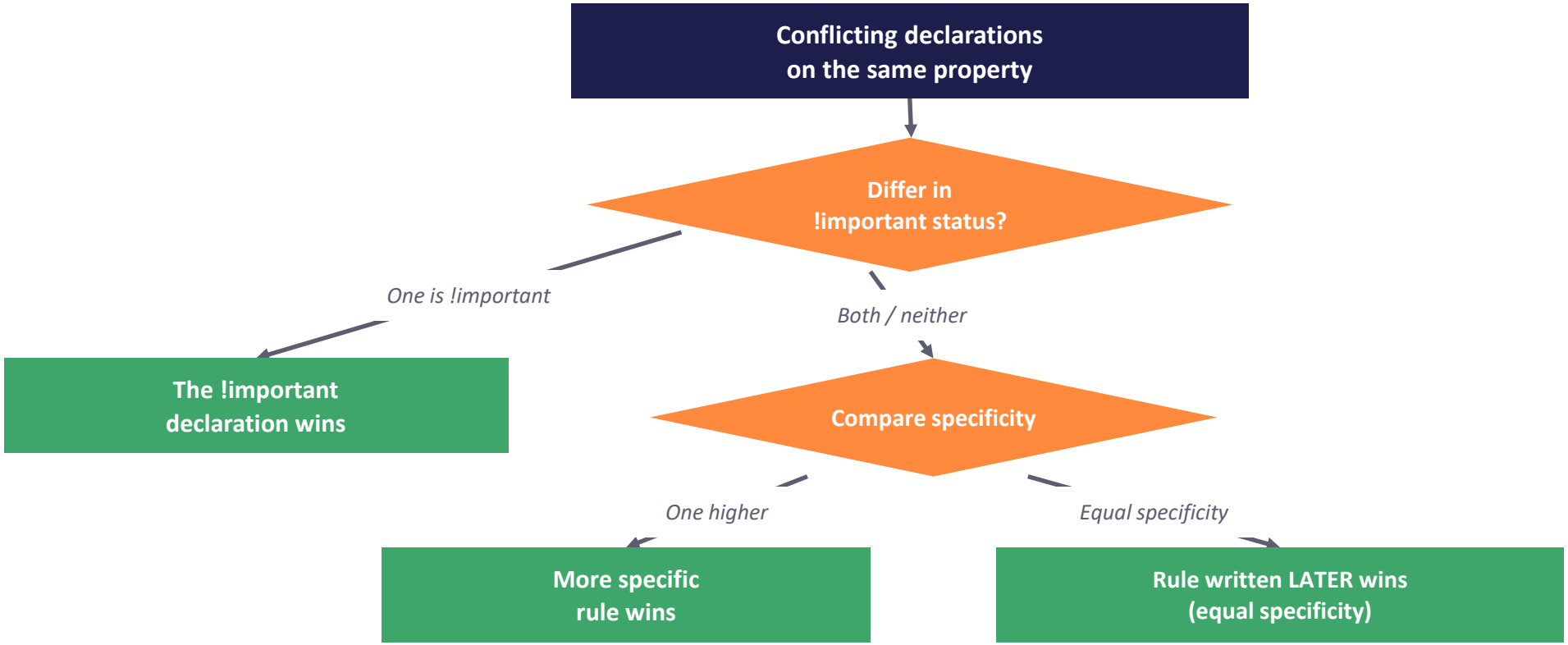
What colour am I?

EVERY CONFLICT CASE

How the Browser Resolves Conflicts

Case	The Winner
1. Different specificity	The more specific rule, regardless of order
2. Equal specificity	The rule written LATER in the source
3. One rule is !important	The !important rule, regardless of specificity
4. Both rules !important	Falls back to specificity, then order
5. Inline style vs. stylesheet	The inline style — unless the stylesheet rule is !important

The Resolution Flowchart



Every cascade conflict resolves through exactly these branches, in this order — importance first, then specificity, then source order.

Case 5 in Code

```
<p id="lead" style="color: purple;">...</p>

#lead { color: blue; }              /* loses to inline */
#lead { color: green !important; } /* beats inline   */
```

Without the !important line, the paragraph is purple. With it, green — one of the few times !important is the right tool.

The 4-Part Specificity Score

Column	Counts
Inline	1 if the style is inline, else 0
IDs	Number of ID selectors
Classes / attrs / pseudo-classes	Number of these
Elements / pseudo-elements	Number of these

#nav .item a:hover -> (0,1,2,1) beats nav ul li a -> (0,0,0,4) — one ID always outweighs any number of elements, no matter how long the other selector looks.

How Specificity Is Calculated

Part of the Selector	Category	Column It Counts Toward
#nav	ID selector	IDs: +1
.item	Class selector	Classes / attrs / pseudo-classes: +1
:hover	Pseudo-class	Classes / attrs / pseudo-classes: +1
a	Element selector	Elements / pseudo-elements: +1

#nav .item a:hover totals to (0 inline, 1 ID, 2 classes/pseudo, 1 element) = (0,1,2,1). Every part of nav ul li a is a plain element, so it totals to (0,0,0,4). Compare column by column, left to right: the IDs column differs first (1 vs. 0), so #nav .item a:hover wins outright — the comparison never even reaches the Elements column.

An Example Where Inheritance Does NOT Happen

```
.parent {
  border: 4px solid crimson;
  padding: 20px;
  color: darkblue;
}

<div class="parent">
  Parent text is dark blue.
  <p>Child paragraph is dark blue too.</p>
</div>
```

color inherits, so the <p> is dark blue too. border and padding do NOT inherit — only .parent itself gets the crimson outline and padding.

Inherited vs. Not Inherited

Inherited by Default	Not Inherited by Default
color	margin
font-family, font-size, font-weight	padding
line-height	border
text-align	width, height
visibility	background / background-color

Five Ways to Write a Colour

```
.a { color: red; }                  /* named colour */
.b { color: #ff0000; }              /* hex code */
.c { color: rgb(255, 0, 0); }       /* red, green, blue */
.d { color: rgba(255, 0, 0, 0.5); } /* rgb + alpha */
.e { color: hsl(0, 100%, 50%); }    /* hue, sat, lightness */
```

Red text (named colour: red)

Red text (hex code: #ff0000)

Red text (rgb(255, 0, 0))

Red text (rgba(255, 0, 0, 0.5) - 50% transparent)

Red text (hsl(0, 100%, 50%))

Font Properties

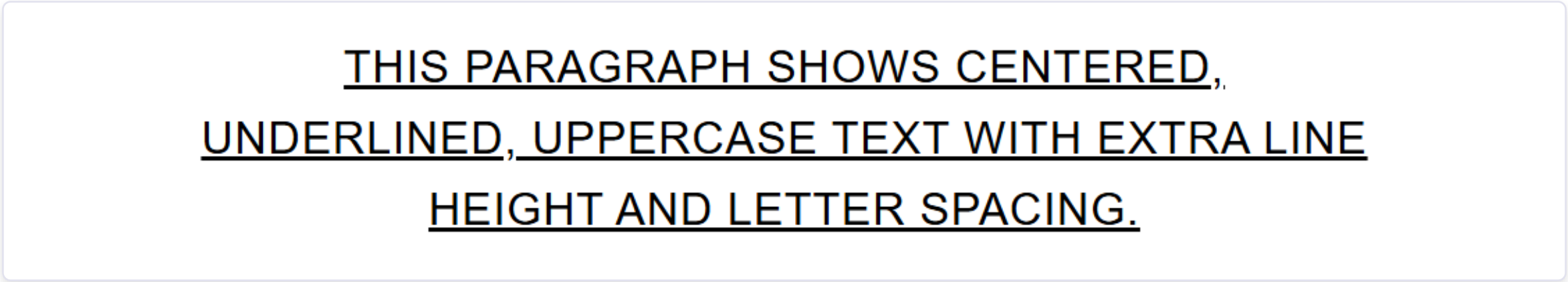
font-family (a fallback list), font-size, font-weight, font-style — combined via the font shorthand too.

```
p {
  font-family: "Helvetica Neue", Arial, sans-serif;
  font-size: 16px;
  font-weight: bold;
  font-style: italic;
}
```

This paragraph demonstrates font-family, font-size, font-weight, and font-style together.

Text Properties

```
p {
  text-align: center;
  text-decoration: underline;
  text-transform: uppercase;
  line-height: 1.6;
  letter-spacing: 0.5px;
}
```



Background Properties

```
.card {
  background-color: #f5f5f5;
  background-image: url("pattern.png");
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
}
```

A background shorthand combines several of these into one declaration, just like font does.

KEY TAKEAWAYS

Lecture 5 in Six Points

1. CSS separates presentation from content; external style sheets are preferred for real projects.
2. Selectors decide what gets styled: element, class, id, attribute, grouping, descendant.
3. The cascade resolves conflicts in order: importance, then specificity, then source order — every time.
4. Specificity is a 4-part score compared column by column; one ID beats any number of classes or elements.
5. !important should be a last resort — it breaks normal specificity reasoning.
6. Text properties like color inherit from parent to child; box properties like border and margin do not.