COSI 152A

Web Application Development

CSS style levels

Style levels

- The Cascading nature of CSS indicates that there are multiple levels of style sheets.
- Specific styles overwrite general styles.
  - A style can be more specific by using:

    - class selectors (groups of elements) and
  - Even more specific by using:

    - id selectors (individual elements).

- Life is found in layers.

Body styles

- Write a selector for the body element to apply the style to the entire body of your page.
- Saves you from manually applying a style to each element

```
body {
 font-size: 16px;
}
```

Inheriting styles

- Styles get inherited from containing elements
    - Not all properties are inherited (notice the link's color)

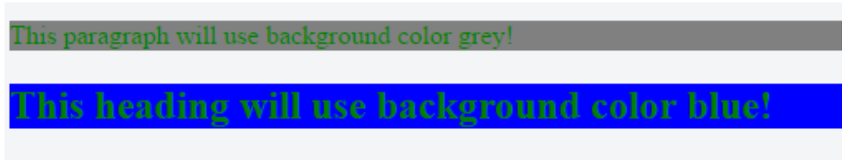
```
body {
  font-family: sans-serif;
  background-color: pink;
}
p {
  color: green;
}
a {
  text-decoration: underline;
}
h2 {
  font-weight: bold;
  text-align: center;
}
```

Styles that conflicts

- When two styles set conflicting values for the same property, the latter style takes precedence.

```
/* select multiple elements separated by commas */
p, h1, h2 { color:
  green;
  background-color: grey;
}
h2 {
  background-color: blue;
}

<p> This paragraph will use background color grey! </p>
<h2> This heading will use background color blue! </h2>
```


This paragraph will use background color grey!

This heading will use background color blue!

Order of styles

- It's called Cascading Style Sheets because the properties of an element cascade together in the following order:
  - Browser's default styles (reference)
  - External style sheet files (in a <link> tag)
  - Internal style sheets (in a <style> tag in the page header)
  - Inline style (the style attribute of an HTML element)
- Cascading works from top to bottom inside the page.
  - Depends on the order – later styles will always override top ones.

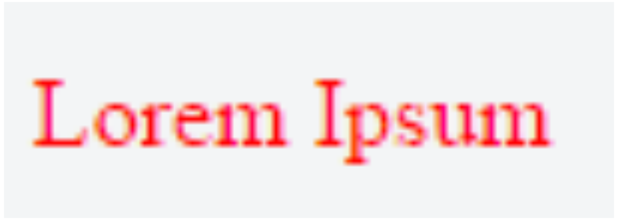
Inheritance vs. cascading

- Inheritance is how elements in the HTML markup inherit properties from their parent elements
- Cascading is how different CSS rule sets are applied to HTML elements, and how conflicting rules do or don’t override each other.

Which rule wins?

```
<p  class="RedColor BlueColor"> Lorem Ipsum </p>
```

```
#YelloColor { color:     yellow;
}
.BlueColor { color:      blue;
}
.RedColor { color:       red;
}
```


Lorem Ipsum

Style Specificity

- When multiple styles are applied to an element and have the same origin precedence:
  - The most specific one is applied.
  - If they have the same specificity, then the later one will be used.

```
<p> <q><em id="recent" class="awesome">Which awesome color?</em></q></p>

em#recent.awesome { color: orange; }
p { color: gray; }
q { color: green; }
em { color: yellow; }
.awesome { color: blue; }
em.awesome { color: red; }
#recent { color: black; }
```

"*Which awesome color?*"

"*Which awesome color?*"

"*Which awesome color?*"

"*Which awesome color?*"

"*Which awesome color?*"

"*Which awesome color?*"

"*Which awesome color?*"

Specificity and conflicts

- Specificity: decide which one should win when two or more rules conflict.
- Rules: each rule's overall selector is given a score based on the following rules and the rule with highest score wins if there's a conflict:
  - Any HTML element mentioned in the rule scores 1 point
  - Any class mentioned in the rule scores 10 points
  - Any ID mentioned in the rule scores 100 points
- Examples:
  - p.banner - 11
  - div.box > p - 12
  - body #logo .box p.banner - 122

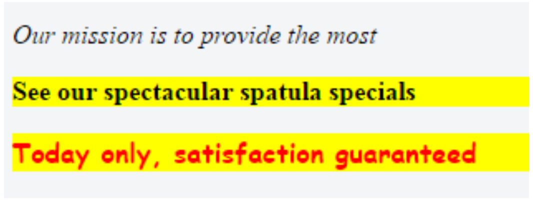
The HTML class and id attribute

- **id** attribute allows you to give a unique ID to any element on a page
  - Each ID must be unique
  - Can only be used once in the page
- **class** attribute is used to group elements and give a style to only that group
  - a class can be reused as much as you like on the page

class vs. id

```
<p id="mission">Our mission is to provide the most</p>
<p class="special">See our spectacular spatula specials</p>
<p class="special shout">Today only, satisfaction guaranteed</p>
```

```
#mission { /* the element with id="mission" */
  font-style: italic;
  color: #000000;
}
.special { /* any element with class="special" */
  background-color: yellow;
  font-weight: bold;
}
p.shout { /* only p elements with class="shout" */
  color: red;
  font-family: cursive;
}
```



Our mission is to provide the most

See our spectacular spatula specials

Today only, satisfaction guaranteed

pseudo-classes and pseudo-elements

- A pseudo-class is used to define a special state of an element
    - Style an element when a user mouse’s over it
    - Style visited and unvisited links differently
    - Style an element when it gets focus
- A CSS pseudo-element is used to style specified parts of an element
    - Style the first letter, or line, of an element

      ::first-line, ::first-letter
    - Insert content (pseudo element) before, or after, the content of an element

      ::before, ::after

```
selector:pseudo-class { property:value; } /* single colon */
selector::pseudo-element { property:value; }  /* double colon */
```

pseudo-classes and pseudo-elements

class	description
:active	an activated or selected element
:focus	an element that has the keyboard focus
:hover	an element that has the mouse over it
:link	a link that has not been visited
:visited	a link that has already been visited
:not(selector)	all elements that do not match the given CSS selector
::first-line	the first line of text inside an element
::first-letter	the first letter of text inside an element

Examples pseudo-classes

```
/* unvisited link */
a:link { color: #FF0000; }

/* visited link */
a:visited { color: #00FF00; }

/* mouse over link */
a:hover { color: #FF00FF; }

/* click on a link */
a:active { color: #0000FF; }
```

More info and examples: Pseudo-classes and Pseudo-elements

Thank You!