

CS 2033

Multimedia & Communications II

LECTURE 4 – CASCADING STYLE SHEETS (CSS)

CSS

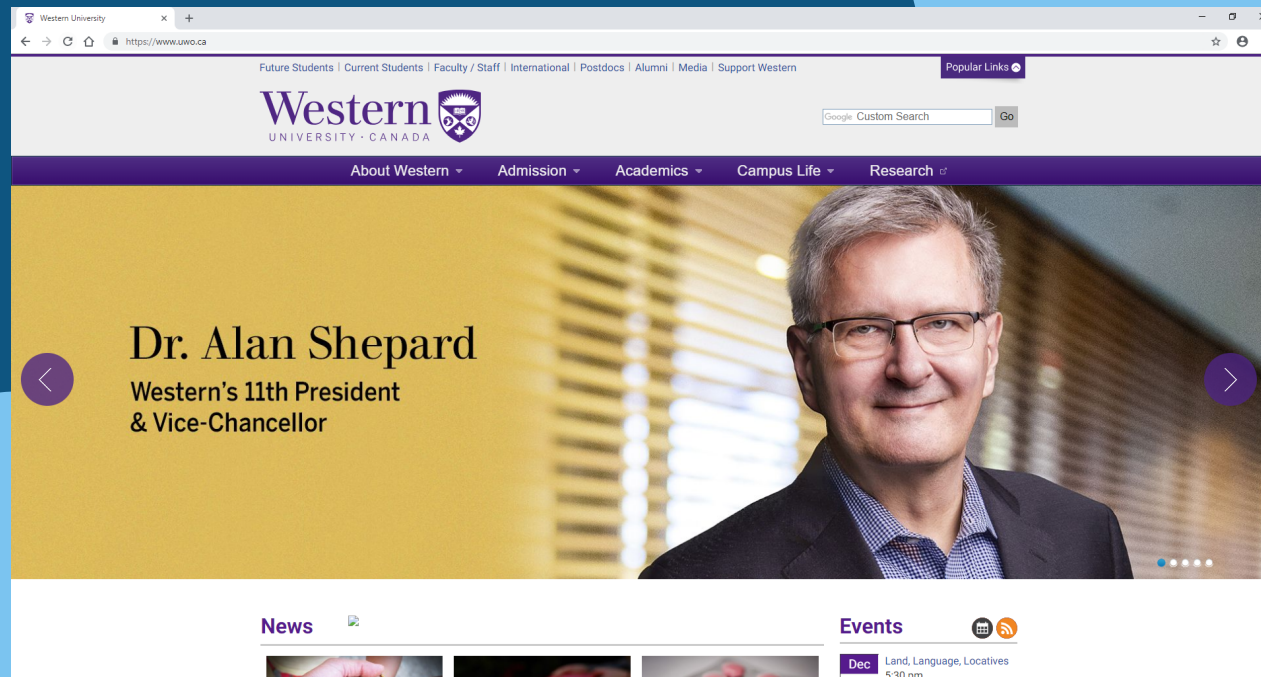
2

- ▶ Cascading Style Sheets
- ▶ CSS is used only to style websites.
- ▶ This is the standard for styling and goes hand in hand with HTML.
- ▶ Used for layouts/positioning of elements and their appearance, like colour, font colour, border, etc.

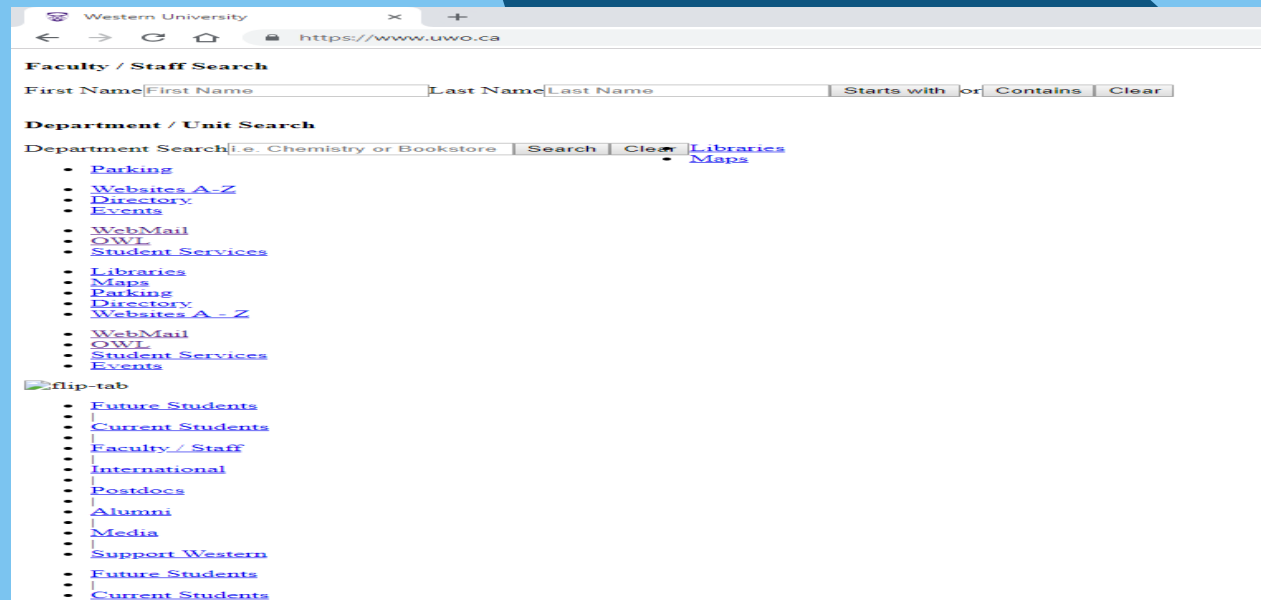
CSS

3

With CSS



Without
CSS



CSS

- ▶ Remember that divs and many other HTML elements can be nested within one another.
- ▶ This is helpful for creating layouts.
- ▶ This relationship is known as parent-child, where the parent is the container / outer element and the child is the inner element.

element 1 {
 element 2
}

- ▶ What styles can be applied in CSS?
 - ▶ Tons! You'll need to know the common ones but not all of them.
- ▶ How are they applied?
 - ▶ 3 main ways to apply the basic styles (more for advanced styles).
- ▶ Where does CSS code go?
 - ▶ 3 different placement options.

Style types

6

► Layout - *structure*.

- Width, height
- Position type
- Position values
- Display type / float
- Margins and padding
- Top, right, bottom, and left (I call these TRBL or "trouble" ☹)

Style types

7

► Appearance

- Background colour
- Background image/texture
- Font colour
- Border style
- Rounded corners
- Opacity



Style types

8

- ▶ Many styles overlap both categories but I try to label them by their primary function.
 - ▶ i.e. size is used for layout but also impacts the appearance.
- ▶ Some styles only work if other styles are set in a certain way.
 - ▶ You will see this very soon when we discuss positioning.

Layout-based styles

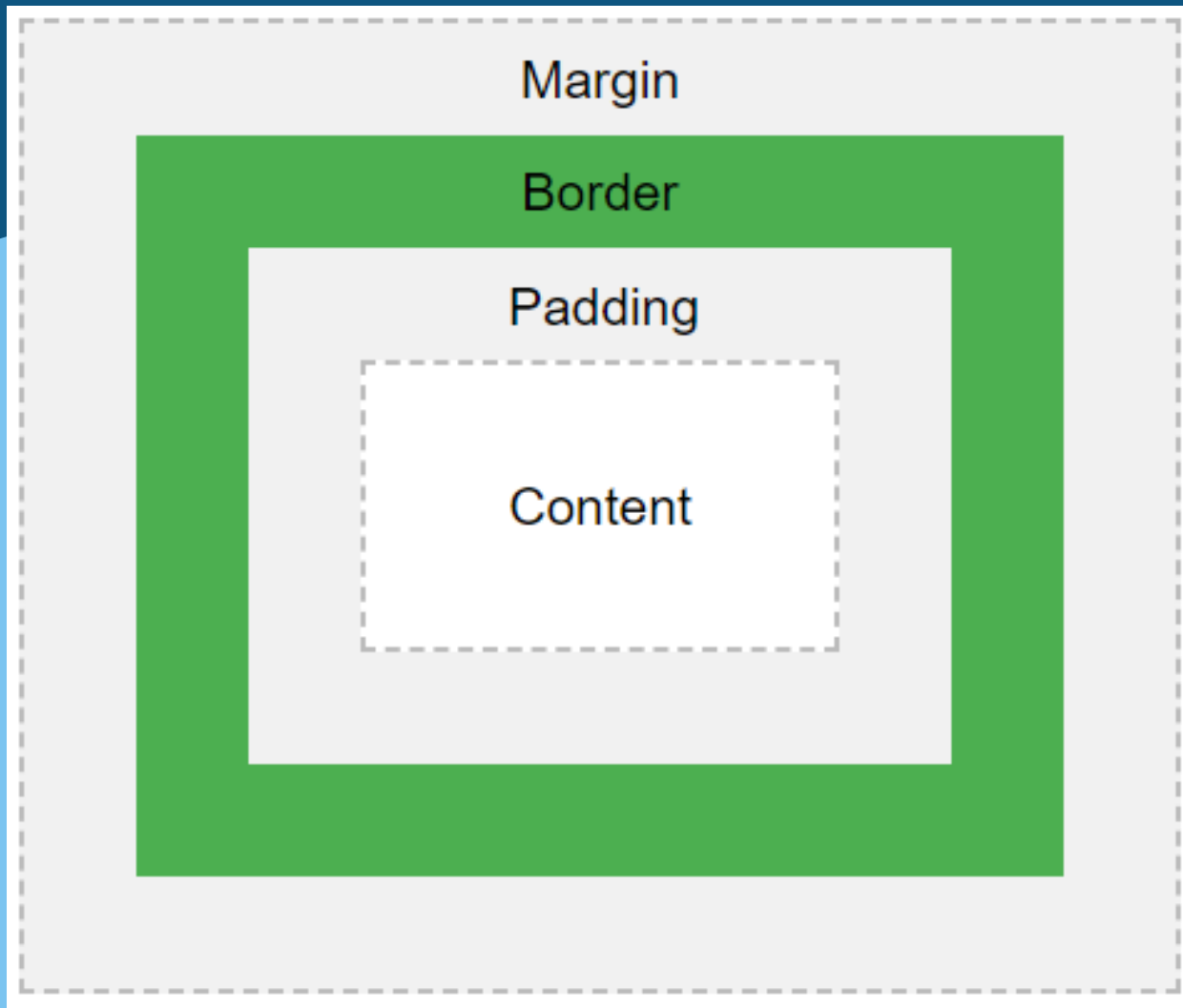
9

- ▶ **Width** and **height** are simple. - px
- ▶ **Padding** is the space just inside the element, keeping its contents away from the edge. 
- ▶ **Margin** is the space outside the element, keeping it away from other elements. 
- ▶ Most size styles are in px or %.

Layout-based styles

10

Box - side model.



Layout-based styles

11

- ▶ Positioning elements can be done a few ways (or a combination).
- ▶ By default, elements are added sequentially top to bottom.
- ▶ Depending on size and layout styles, they may be added left to right in a row too. *images if it fits.*

Layout-based styles

12

- ▶ The **position** style type determines how (but not where!) the element is positioned in the page or its parent.
- ▶ The default value is static, meaning it's added sequentially in the site and cannot be moved.

Layout-based styles

13

► Other **position** options are:

► Relative – similar to static but can be shifted with TRBL values.

Top Right Bottom Left.

► Absolute – location is directly based on TRBL values **within its parent!**

relative to its parent

► Fixed – location is locked in place.

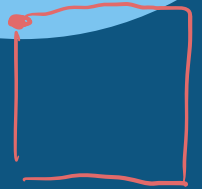
menu stays at the top.

► Sticky – position changes between relative and fixed on scrolling.

Layout-based styles

14

- ▶ Top, right, bottom, and left (TRBL) can be set, but their behavior depends on the position type.
 - ▶ No effect on static position.
 - ▶ Think of this as a Cartesian plane grid, with the top-left corner being (0,0) in terms of (left,top).
 - ▶ You can start from any corner though! Use either T or B, and either L or R.



Layout-based styles

15

- ▶ In addition to position, another way to affect layout is with display.
- ▶ There are several possible values for this but the most important ones for now are:
 - ▶ Block – takes up entire row.
 - ▶ Inline-block – can be placed in row.
 - ▶ None – not added to page.

images by default

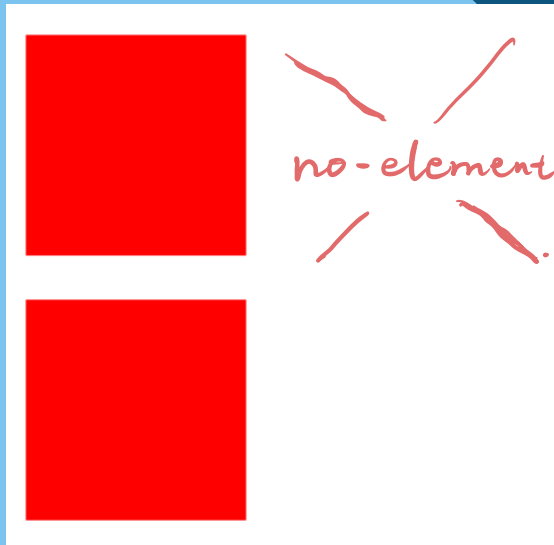
Hide contents.

JavaScript could control content to hide or not.

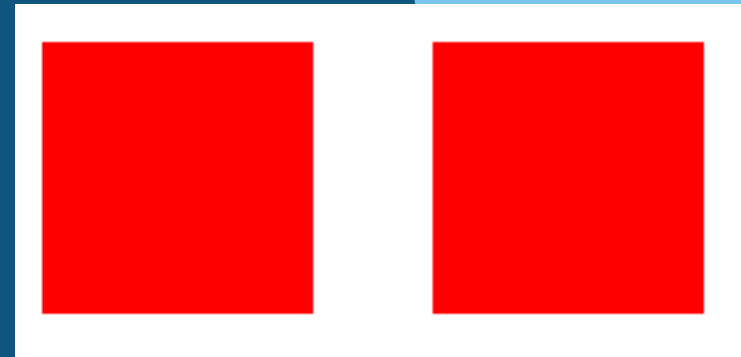
Layout-based styles

16

- If you want to place elements side by side, then try inline-block.



Block display



Inline-block display

Layout-based styles

17

- ▶ Positioning and creating layouts with CSS can be complex.
- ▶ We will discuss this topic in more detail next week.
- ▶ For now we are still going through the basics of CSS styles.

Appearance-based styles

18

RGB/color name.

- ▶ Background can be a solid colour, gradient, transparent, or an image.
- ▶ For a texture/image, you can also set whether it should be repeated (tiled), its size, and position.

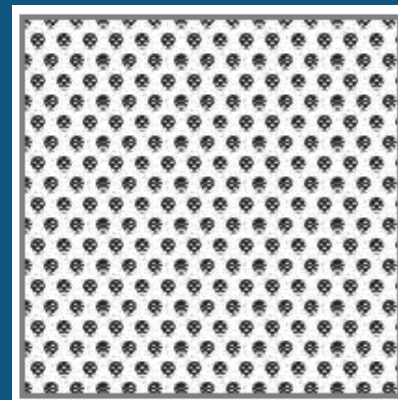
tiled



horizontally



vertically



no-repeat.

photographs don't tile good

Don't do it!

Appearance-based styles

19

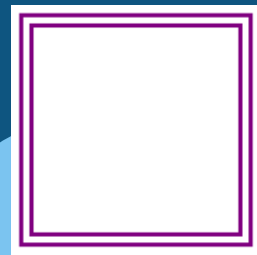
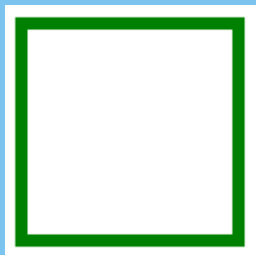
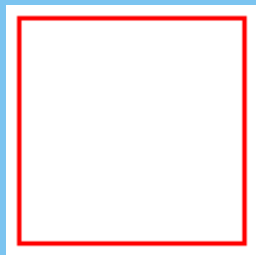
- ▶ Font colour can be any solid color.

- ▶ Border styles have 3 parts:

- ▶ Colour

- ▶ Width

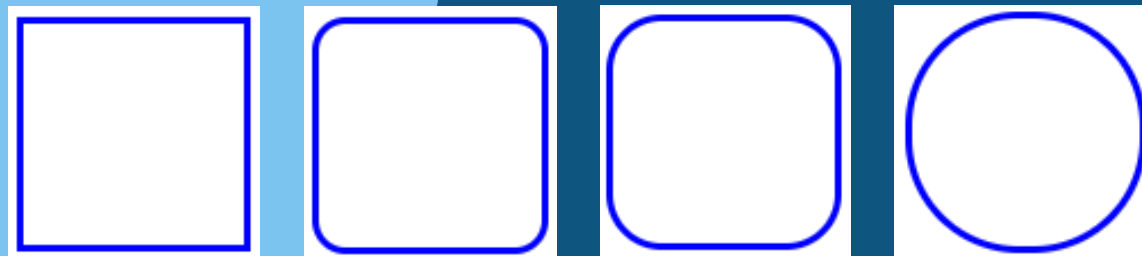
- ▶ Line type (solid, dotted, etc.)



Appearance-based styles

20

- ▶ Corners can be square or rounded.
- ▶ Rounding values are usually in px.
- ▶ If you use really high rounding values, you can create a circle [?]
 - ▶ If width = height and the rounding value is at least half of that width.



Colours

21

- ▶ Several style types are based on colour (i.e. background colour, font colour, border colour).
- ▶ They can be hexadecimal or RGB codes.
A lot of popular colours can also be called by name! transparent is another option.
- ▶ <http://www.html-color-names.com/color-chart.php>

Style examples

22

▶ How are styles actually set?

▶ *property: value;*

▶ Examples:

▶ width: 200px;

▶ display: block;

▶ position: absolute;

Style examples

23

► Colour examples:

- background-color: beige; *pre name*
- background-color: transparent;
- background-color: #9595AA *hex*
- *font color.* color: yellow;
- color: rgb(255,32,175)
- What is *font color* color vs. background-color?

Style examples

24

▶ Border examples:

- ▶ border: solid 2px darkred; *order doesn't matter.*
 - ▶ border: rgb(0,0,50) dotted 1px;
 - ▶ border-bottom: solid 1px #A744B9;
 - ▶ border-width: 4px;
 - ▶ border-top-color: purple;
- ▶ Lots of flexibility with borders!

Rule-sets

25

- ▶ Now that we know how to make individual styles, how do we group them and apply them?
- ▶ A **rule-set** is a group of styles for a certain selector or selectors.
- ▶ There can be any number of styles within a rule-set.

Selectors

26

- ▶ Wait, what are selectors?
- ▶ Selectors are ways of determining which element(s) are given the styles of the rule-sets. *what to apply these styles*
- ▶ We're essentially indicating which element(s) we want to apply a style to and then using any combination of rules to create the overall style.

Selectors

27

- ▶ Rule-sets are formatted like:

- ▶ *selector {
 property1: value1;
 property2: value2;
 ...
}*

- ▶ This will make more sense when you see the selectors.

- ▶ *property1: value1* represents a generic property-value style rule.

Selectors

28

- ▶ There are 3 main types of selectors.
 - ▶ Based on tag / element type.
 - ▶ Based on class name.
 - ▶ Based on ID.
- ▶ There are additional types based on the state of the element, known as pseudo-classes.

Tag selectors

29

- ▶ Tag selectors apply to **all** elements of the specified HTML tag.
 - ▶ i.e. <p>, <h1>, <body>,
- ▶ These selectors are labelled with the tag name, as it is in HTML, but without the <> brackets.
 - ▶ i.e. p, h1, body, img

```
body {  
  color: blue  
}
```

Tag selectors

30

```
▶ p {  
    color: red;  
}
```

```
▶ body {  
    margin: 0;  
}
```

```
▶ div {  
    background-color: #ff0000;  
}
```

Class selectors

31

- ▶ Class selectors apply to **all** elements that are given the specified class.
- ▶ HTML elements can be given a class as an attribute.
 - ▶ i.e. <div class='myclass'>
- ▶ Classes can be applied to any number of elements and any combination of element types.

Class selectors

32

- ▶ These selectors in CSS are labelled with a period (.) followed by the specific class name it applies to.
this dot indicates it is a class.
 - ▶ i.e. `.myclass`, `.anotherclass`
- ▶ Ensure the class name is spelled identical in HTML and CSS.
 - ▶ i.e. `<div class='my-class'>` will not match the selector `.myclass`

Class selectors

33

```
▶ .my-class {  
    position: relative;  
}
```

```
▶ .nav {  
    margin: 5px;  
    color: darkblue;  
    width: 100%;  
    height: 50px;  
}
```

ID selectors

34

- ▶ ID selectors apply to the element with that ID (if there is one).
- ▶ Just like classes, HTML elements can be given an ID as an attribute.
 - ▶ i.e. `<div id='menu'>`
- ▶ Unlike classes, IDs must be **unique** and not given to multiple elements.
 - ▶ This is important!

ID selectors

35

- ▶ These selectors in CSS are labelled with a pound sign (#) followed by the specific ID name it applies to.
 - ▶ i.e. #menu, #profile-picture
- ▶ Ensure the ID name is spelled identical in HTML and CSS.
 - ▶ i.e. <div id='topmenu'> will not match the selector #menu

ID selectors

36

```
▶ #menu {  
    height: 100px;  
    line-height: 100px;  
    background-color: black;  
    color: white;  
}  
  
▶ #title {  
    font-size: 40px;  
}
```

Styling web forms

37

- ▶ There are many form input types.
- ▶ How can we apply a style to all/several inputs at once, or all inputs of a certain type at once?
e.g. type = "text"
- ▶ input { } applies to all "input" tags.
- ▶ Textarea is not made with an input tag so it will not be affected. [?]
- ▶ Use textarea { } for these fields.

Styling web forms

38

▶ i.e. make all form fields 200px wide.

```
▶ input {  
    width: 200px;  
}
```

```
▶ textarea {  
    width: 200px;  
}
```

Styling web forms

39

- ▶ If we want the same styles applied to "inputs" and "textareas", it would be redundant to have two identical rule-sets for the two types.
- ▶ Selectors can be grouped together using a comma to separate them.
- ▶ *group together.*
input, textarea {
 width: 200px;
}

Styling web forms

40

- ▶ Grouping rule-sets is not only for form input elements but for any combination of selectors (tags, classes, and IDs).
- ▶ p, input, ^{class}.longtext, ^{ID}#login {
width: 200px;
color: blue;
}

Styling web forms

41

- ▶ How about if we want to style one specific type of input field?
- ▶ CSS allows us to select based on attribute values as well!
- ▶ Remember most input types are specified by the **type** attribute.
 - ▶ i.e. `<input type="text" />`
 - ▶ i.e. `<input type="radio" />`

Styling web forms

42

- ▶ We can specify an attribute value in square brackets [] to select that type for a CSS selector.

- ▶ `input[type=text] {
 border: solid 2px #FA4949
}`

- ▶ `input[type=submit] {
 width: 200px;
}`

button.

Styling web forms

43

- ▶ These can also be incorporated in grouped selectors.

- ▶ `input[type=submit], #title, p {
 color:red;
}`

- ▶ `input[type=text], textarea {
 font-size: 20px;
}`

Adding CSS in webpages

44

- ▶ There are 3 ways of adding CSS to webpages:
 - ▶ Inline – in HTML element attributes.
 - ▶ Internal – in HTML head section.
 - ▶ External – in its own file.

Inline CSS

45

- ▶ One way to add CSS is directly in HTML tags in the **style** attribute.
- ▶ This can work well for applying a style to a single element and doing so quickly for testing purposes.
- ▶ i.e. `<div style='width: 50%; height:300px'>Welcome</div>`

Internal CSS

46

- ▶ Inline CSS is generally not a good option since it only applies to one element.
- ▶ To apply styles to an entire page, you can add rule-sets into the head section of the HTML.
- ▶ CSS is meta data!
attach to webpage.

Internal CSS

47

- ▶ Within the head, use the <style> tag to create a place for CSS and then add the styles in there.
- ▶ Works well for a single page site, or styles that only apply to one page.
- ▶ Definitely better than inline styles.

Internal CSS

single-page website

48

```
<head>
```

```
<style>
```

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

```
  div {
```

```
    width: 300px;
```

```
    border: solid 2px red;
```

```
  }
```

```
</style>
```

```
</head>
```


External CSS

49

- ▶ However, internal CSS is still not completely efficient.
- ▶ Suppose you have a website with multiple pages and want the styles to apply to all pages.
- ▶ The best option is **external** CSS.
- ▶ Store the CSS in its own file(s) and link the webpages to the CSS file(s).

External CSS

50

- ▶ External CSS is stored in files with the .css extension.
- ▶ Linking these files into HTML pages is very simple:
 - ▶ <link rel="stylesheet" type="text/css"
href="styles.css">
 - ▶ This is also meta information so it goes in the head section of the HTML.

External CSS

51

index.html

```
<html>
<head>
  <link rel='stylesheet' type='text/css' href='styles.css'>
</head>
<body>
```

styles.css

```
body {
  margin:0;
  padding:0;
  background-color:cyan;
}

p {
  color:yellow;
}

#main-title {
  font-size:45px;
}
```

CSS rules

52

- ▶ Styles are applied in top-to-bottom order generally. *left-to-right.*
- ▶ This only matters if there are conflicting rules or rule-sets.
- ▶ The order doesn't matter otherwise.
- ▶ `p {color: red; width: 50px; }` is the same as `p {width: 50px; color: red; }` since the rules are independent.

CSS rules

53

- ▶ So where does the order matter?
 - ▶ Conflicting rules within a rule-set.
 - ▶ Multiple rule-sets with conflicting styles applied to an element.
- ▶ Let's look at examples of each.

CSS rules

54

- ▶ In cases of conflicting rules within a rule-set, the bottom-most rule overrides previous ones.
- ▶

```
p {  
  color: red;  
  color: blue;  
}
```
- ▶ In this case, **color: blue** is applied only. It overrides color: red;

@ Override

CSS rules

55

- ▶ When multiple rule-sets are applied, it's a little more complicated.
- ▶ `p { color: red; }`
`.home { color: blue; }`
- ▶ `<p class='home'>Hello world</p>`
- ▶ Does the text turn red because it's a paragraph or blue because it has the 'home' class applied to it?

CSS rules

56

- ▶ The class rule-set takes precedence so the text will be blue.
- ▶ How about if there is an ID too?
- ▶

```
p { color: red; }  
.home { color: blue; }  
#title {color: green; }
```
- ▶

```
<p class='home' id='title'>Hello  
world</p>
```


CSS rules

57

- ▶ The ID rule-set will be applied so the text will be green.
- ▶ Why does this happen?
- ▶ CSS rules are assigned a **specificity** or a priority weighting to indicate the precedence in cases of conflicting rules or rule-sets.

CSS rules

58

► The specificity order (low to high) is:

1. ③ Type selectors (p, div, etc.)
2. ② Class selectors (.home, etc.)
3. ① ID selectors (#title)

► This is why class overrides type, and ID overrides both type and class in our examples.

CSS rules

59

- ▶ There's a way to break the regular order of rule-set specificity.
- ▶ The word important immediately after a style gives it top priority.
 - ▶

```
p {  
    font-size:24px important;  
}
```

override all other styles.
- ▶ It's not recommended to use this unless you absolutely need to.

Design tips

60

- ▶ Tips on smart website design:
 - ▶ Use web-safe fonts or Google Fonts.
 - ▶ Create a consistent and cohesive design for your website.
 - ▶ Limit the number of colours you use.
 - ▶ Ensure all text is readable.
 - ▶ Avoid having tons of text.
 - ▶ Do not center paragraphs of text.

Hard to read.

Additional tips

61

- ▶ While creating a website, use flashy backgrounds or borders to help see where elements start and end. I often use reds and yellows to help with this.
- ▶ Once they are in the correct place, revert them to the colours you desire.

Additional tips

62

- ▶ Sometimes you will change CSS but the change is not displayed when you refresh the browser.
- ▶ Might be due to caching. Browsers save website information so that it can load quicker the next time.
- ▶ To get around this, close Chrome and then open it in Incognito mode.

