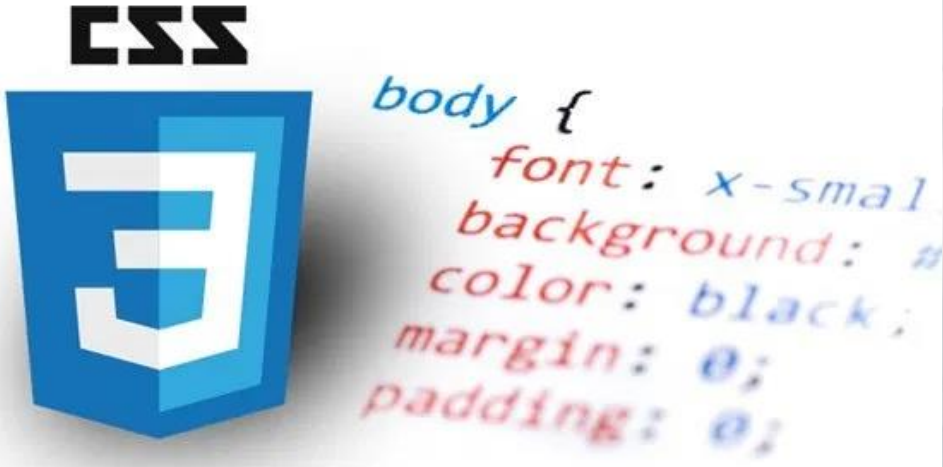
Front-end Development

Lecturer: Ung Văn Giàu
Email: giau.ung@eiu.edu.vn



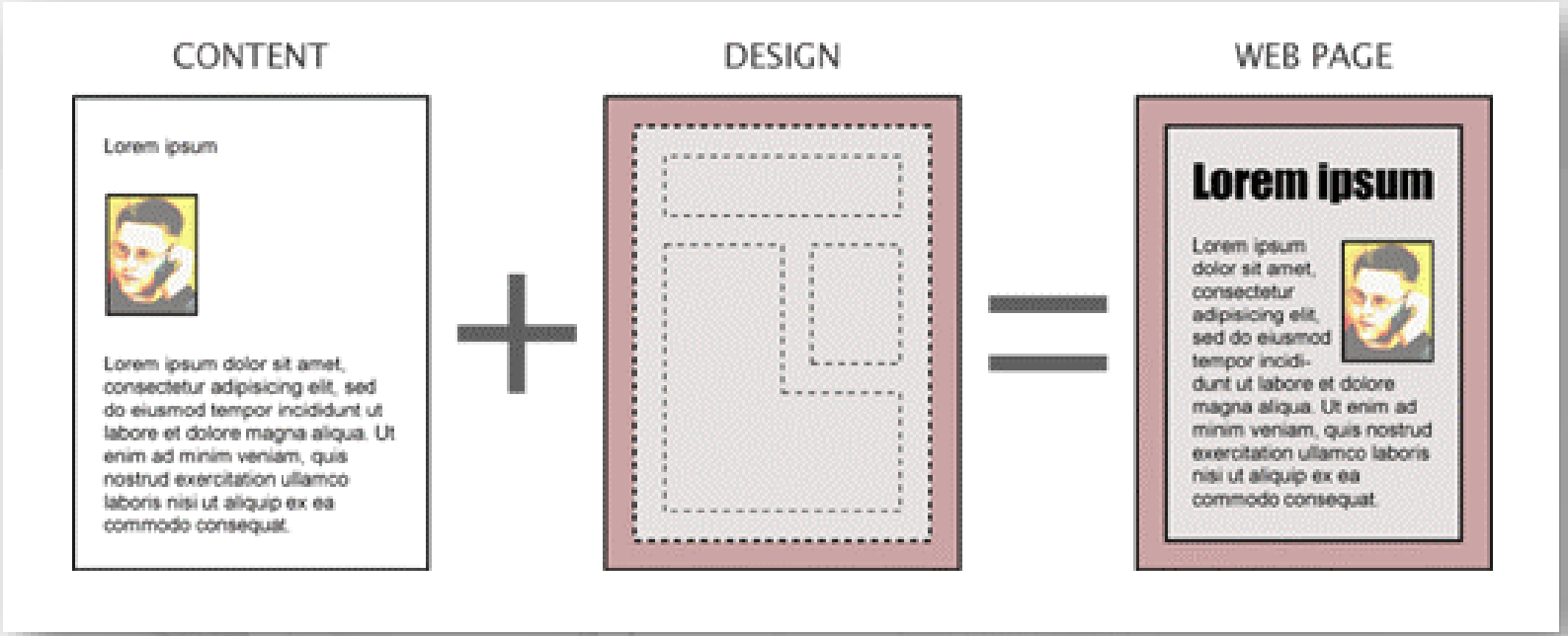
CSS Overview

Cascading Style Sheets

Contents

CSS: A New Philosophy

Separate content from presentation!



Cascading Style Sheets

Separating Content from Presentation

1. CSS Introduction

Styling with Cascading Stylesheets

CSS Introduction

- Cascading Style Sheets (CSS)
  - Used to **describe** the presentation of documents
  - Define **sizes, spacing, fonts, colors, layout**,…
  - Improve content **accessibility**
  - Improve **flexibility**
- Designed to separate presentation from content
- Due to CSS, all **HTML presentation** tags and **attributes** are **deprecated**, e.g., font, center, etc.

CSS Introduction

- CSS can be applied to any XML document

  Not just to HTML / XHTML

- CSS can specify different styles for **different media**
  - On-screen
  - In print
  - Handheld, projection,…
  - … even by voice or Braille-based reader

Why “Cascading”?

Priority scheme determining which style rules apply to element

- **Cascade priorities** or **specificity (weight)** are calculated and assigned to the rules
- **Child elements** in the HTML DOM tree **inherit styles** from their parent
  - Can **override** them
  - Control via **!important** rule

Style Inheritance

Some CSS styles are inherited, and some are not

- **Text-related** and **list-related** properties are **inherited**: color, font-size, font-family, line-height, text-align, list-style,…
- **Box-related** and **positioning** styles are **not inherited**: width, height, border, margin, padding, position, float,…

Style Sheets Syntax

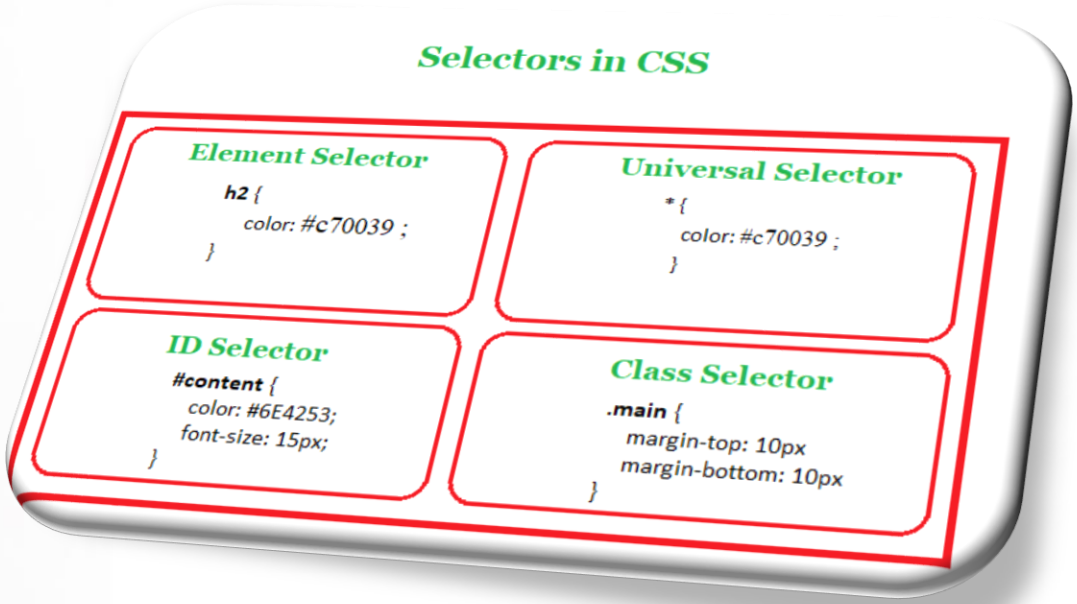
Stylesheets consist of **rules**, **selectors**, **declarations**, **properties** and **values**

- **Selectors** are separated by commas
- **Declarations** are separated by semicolons

  Properties and **values** are separated by colons

```
h1, h2, h3                                  // Selectors
{                                           // Declaration Start
    color           :       green       ;   // Declaration
    font-weight     :       bold        ;   // Declaration
}                                           // Declaration End
```

Properties | Property/Value separator | Values | Declaration separator



2. Common Selectors

Select the Elements to Apply a Style

Selectors

- Selectors **determine** which **element** the rules apply to:
  - **All elements** of specific type (**tag**)
  - Those that match a **specific attribute** (**id, class**)
  - Elements may be matched depending on how they are **nested** in the document tree (HTML)

- Examples:

```
.header a { color: green }
#menu > li { padding-top: 8px }
```

Primary Selectors

- Three primary kinds of selectors:
  - By **tag** (type selector)
  - By element **id**
  - By element **class** name (only for HTML)

```
h1 { font-family: verdana,sans-serif; }
#element_id { color: #ff0000; }
.myClass { border: 1px solid red; }
```

- Selectors can be **combined** with commas:

```
h1, .link, #top-link { font-weight: bold; }
```

Nested Selectors

Match **relative** to element placement:

```
p a { text-decoration: underline; }
```

***** – **universal** selector (avoid or use with care!):

```
p * {color: black}
```

+ selector – used to match "**next sibling**":

```
img + .link { float:right; }
```

Nested Selectors

+ selector

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            div + p {
              background-color: yellow;
            }
        </style>
    </head>

    <body>
        <div>
          <p>Paragraph 1 in the div.</p>
          <p>Paragraph 2 in the div.</p>
        </div>

        <p>Paragraph 3. Not in a div.</p>
        <p>Paragraph 4. Not in a div.</p>
    </body>
</html>
```

Nested Selectors

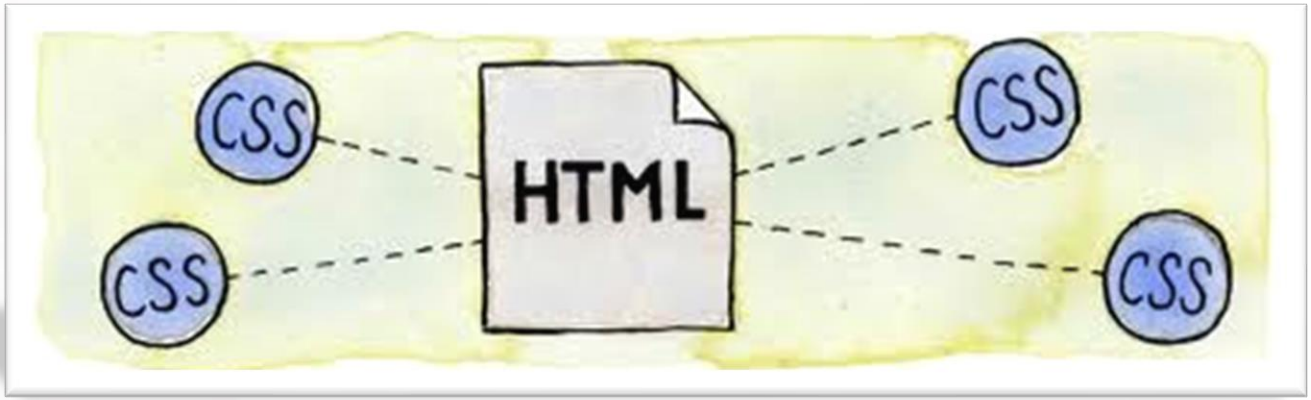
> selector – matches **direct child nodes**:

```
p > .error { font-size: 8px; }
```

.class1.class2 (no space!)

```
p.post-text.special { font-weight: bold; }
```

Matches elements with **both** (all) **classes** applied at the same time



3. Importing CSS Into HTML

How to Use CSS with HTML?

Importing CSS Into HTML

CSS (presentation) can be imported in HTML (content) in **three ways**:

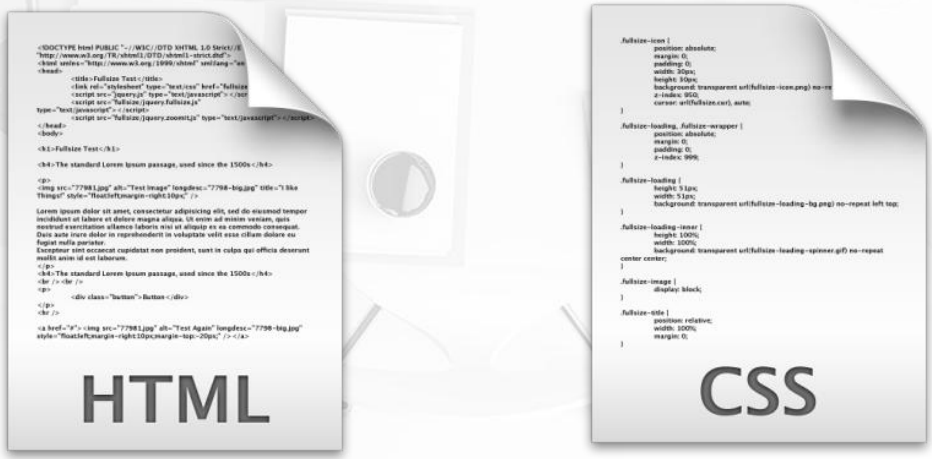
- **Inline:** the CSS rules in the **style** attribute

  No selectors are needed
- **Embedded:** in the **<head>** in a **<style>** tag
- **External:** CSS rules in separate file (best)
  - Usually, a file with **.css** extension
  - Linked via **<link rel="stylesheet" href="…">** tag
  - Via **@import** directive in embedded CSS block

Linking HTML and CSS

Using **external CSS files** is highly **recommended**

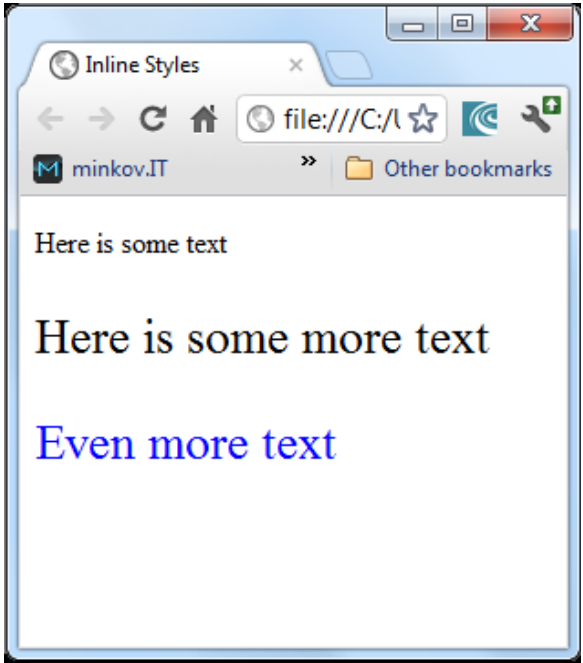
- Simplifies the HTML document
- Improves page **load speed** (CSS file is cached)



Inline Styles

Example

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Inline Styles</title>
    </head>
    <body>
        <p>Here is some text</p>
        <p style="font-size: 20pt;">Here is some more text</p>
        <!--Separate multiple styles with a semicolon-->
        <p style="font-size: 20pt; color: #0000FF;">Even more text</p>
    </body>
</html>
```



Embedded Styles

- Embedded in the HTML in the **<style>** tag:
  - The <style> tag is **placed in the <head>** section of the document
  - **type** attribute specifies the MIME type
    - ✓MIME **describes the format** of the content
    - ✓Other MIME types include text/html, image/gif, text/JavaScript, etc.
    - ✓**Not required in HTML5**

- Used for document-specific styles

```
<style type="text/css"></style>
```

Embedded Styles

Example

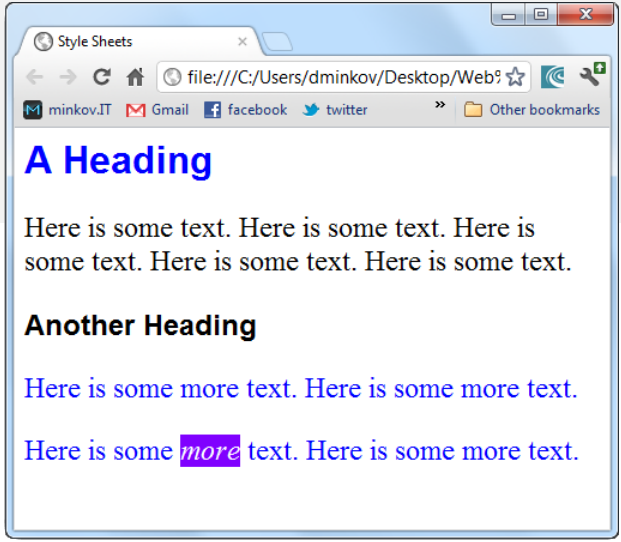
```
<!DOCTYPE html>
<html>
    <head>
        <title>Style Sheets</title>
        <style type="text/css">
            em { background-color: #8000FF; color: white; }
            h1 { font-family: Arial, sans-serif; }
            p  { font-size: 18pt; }
            .blue { color: blue; }
        </style>
    </head>
    <body>
        ...
    </body>
</html>
```

Embedded Styles

Example (cont.)

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Your web title</title>

        <link rel="icon" href="./images/fav.png" type="image/png">
        <!-- Your styles and scripts here -->
    </head>
    <body>
        <!-- Your code here -->
    </body>
</html>
```



External CSS Styles

- External linking
  - **Separate pages** can all use a shared style sheet
  - Only modify a single file to change the styles across your entire Web site

- **link** tag (with a **rel** attribute)
  - Specifies a relationship between current document and another document
  - **link** elements should be in the

```
<link rel="stylesheet" type="text/css" href="styles.css">
```

External CSS Styles

- **@import**
  - Another way to link external CSS files
  - Example:

```
<style type="text/css">
    @import url("styles.css");
    @import "styles.css";
</style>
```

  - Ancient browsers do not recognize @import
  - Use @import in an external CSS file to workaround the IE CSS file limit of 31 files

Summary

Import CSS Into HTML

- **Inline:** the CSS rules in the **style** attribute

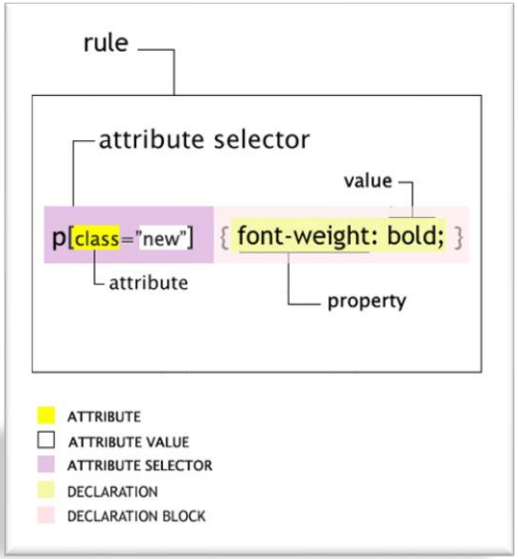
```
<p style="font-size: 20pt;">Here is some more text</p>
```

- **Embedded:** in the **<head>** in a **<style>** tag

```
<head>
    <title>Style Sheets</title>
    <style type="text/css">
        h1 {
            font-family:Arial, sans-serif;
        }
        .content {
            font-size: 18pt;
        }
    </style>
<head>
```

- **External:** CSS rules in separate file (best)

```
<link rel="stylesheet" type="text/css" href="styles.css">
```



4. Attribute Selectors

Picking Elements with Certain Attributes

Attribute Selectors

[ ] selects elements based on attributes

- Element with a **given attribute**

  Selects **<a>** elements with **title**

```
a[title] { color: black; }
```

- Elements with a **concrete** attribute **value**

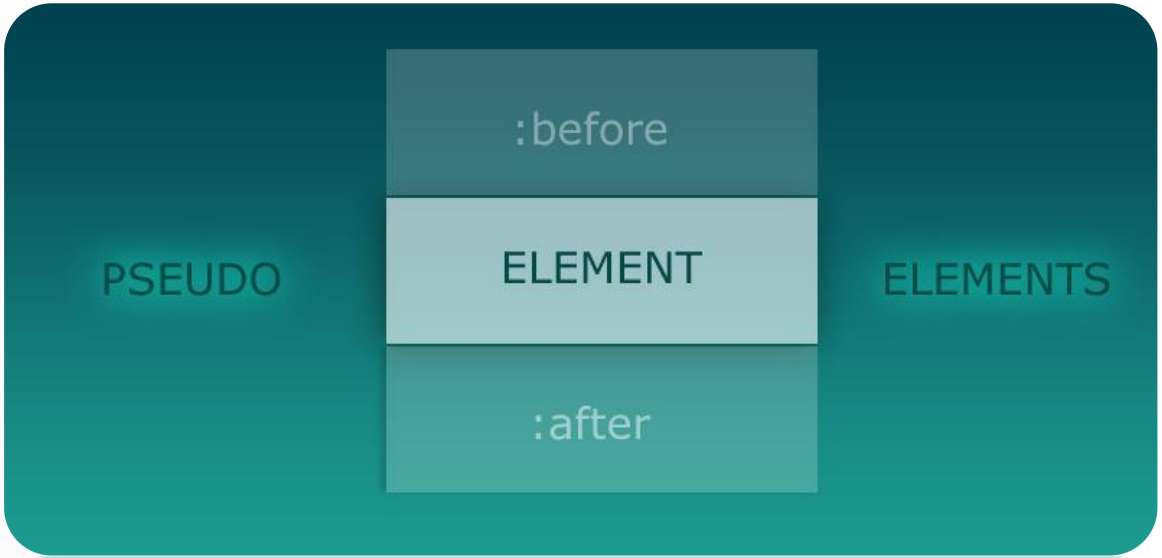
  Selects **<input>** elements with **type="text"**

```
input[type=text] { font-family: Consolas; }
```

- Elements whose attribute **values contain a word**

  Selects **<a>** elements whose title attribute value contains **logo**

```
a[title*=logo] { border: none; }
```



5. Pseudo Selectors

Relative to Element Content or State

Common Pseudo Selectors

- **Pseudo-classes** define **state**

  :hover, :visited, :active , :lang

- **Pseudo-elements** define element “**parts**” or are used to generate content

  :first-line , :before, :after

```
a:hover { color: red; }
p:first-line { text-transform: uppercase; }
.title:before { content: "»"; }
.title:after { content: "«"; }
```

Structural Pseudo-classes

- **E:first-child**
  An E element, first child of its parent
- **E:last-child**
  An E element, last child of its parent
- **E:first-of-type**
  An E element, first sibling of its type
- **E:last-of-type**
  An E element, last sibling of its type

Structural Pseudo-classes

Example

- p:first-child
- div:first-child

```
<div>
    <p>This text is selected!</p>
    <p>This text isn't selected.</p>
</div>

<div>
    <h2>This text isn't selected: it's not a `p`.</h2>
    <p>This text isn't selected.</p>
</div>
```

Structural Pseudo-classes

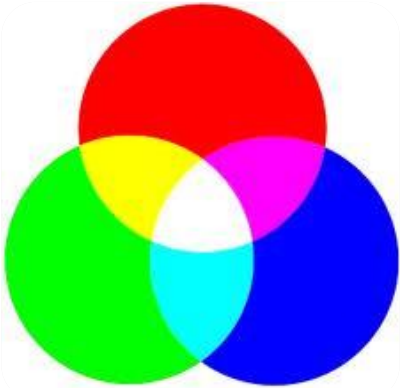
- **E:nth-child(n)**
  An E element, the n-th child of its parent
- **E:nth-of-type(n)**
  An E element, the n-th sibling of its type
- **E:only-child**
  An E element, only child of its parent
- **More detailed descriptions:**
  http://www.w3.org/TR/css3-selectors/#structural-pseudos

The UI Element States Pseudo-Classes

- **E:enabled**
  A user interface element E which is enabled

- **E:disabled**
  A user interface element E which is disabled

- **E:checked**
  A user interface element E which is checked (for instance a radio-button or checkbox)



6. CSS Values

Types, Ranges, Units

CSS Values

- All values in CSS are **strings**
  - They can represent values that are not strings
  - I.e., 14px means size 14 pixels

- **Colors** are set in a red-green-blue format (**RGB**) or a hue-saturation-lightness format (HSL)

  E.g. a RGB color is both in **hex** and **decimal** formats

```
li.nav-item { color: #44f1e1; }
li.nav-item { color: rgb(68, 241, 255); }
```

Size Values

When setting a **size** (width, height, font-size,…) the values are given as **numbers**

- Multiple formats / metrics may be used
  Pixels, ems, e.g., 12px , 1.4em
- Points, inches, centimeters, millimeters
  E.g., 10pt , 1in, 1cm, 1mm
- Percentages, e.g., 50%
  Of the size of the container/font size
- Zero can be used with no unit
  border: 0;

CSS Units

Absolute Lengths

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

CSS Units

Relative Lengths

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

CSS Units

Recommended Use

	Recommended	Occasional use	Not recom-mended
Screen	em, px, %	ex	pt, cm, mm, in, pc
Print	em, cm, mm, in, pt, pc, %	px, ex	

Color Values

Colors in CSS can be represented in few ways

- Using **red-green-blue** or **red-green-blue-alpha**
- Using **hue-saturation-light** or **hue-saturation-light-alpha**

```
color: #f1a2ff;
color: rgb(241, 162, 255);
color: rgba(241, 162, 255, 0.1);

color: hsl(291, 85%, 89%);
color: hsl(291, 85%, 89%, 0.1);
```

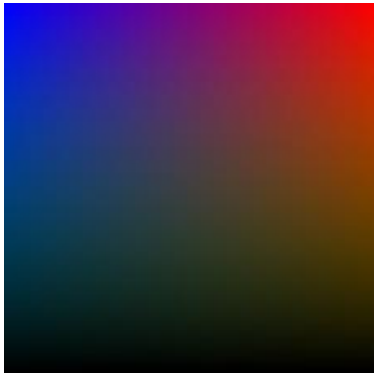
- **Note:**

  Alpha is an **opacity value** (from 0.0 to 1.0)

RGB Colors

- RGB colors are defined with values for red, green and blue intensity
- **Syntax:**
  - **#44fa36** – values are in hex
  - **rgb(<red>, <green>, <blue>)** – decimal values
- The **range** for red, green and blue is **between** integers **0 and 255**

```
color: #07f2b3;
<!- or -->
color: rgb (7, 242, 179);
```



RGBA Colors

- Standard RGB colors with an opacity value for the color (alpha channel)
- **Syntax: rgba(<red>, <green>, <blue>, <alpha>)**
- The range for red, green and blue is between integers 0 and 255
- The range for the **alpha** channel is **between 0.0 and 1.0**
- Example: rgba(255, 0, 0, 0.5)

HSL Colors

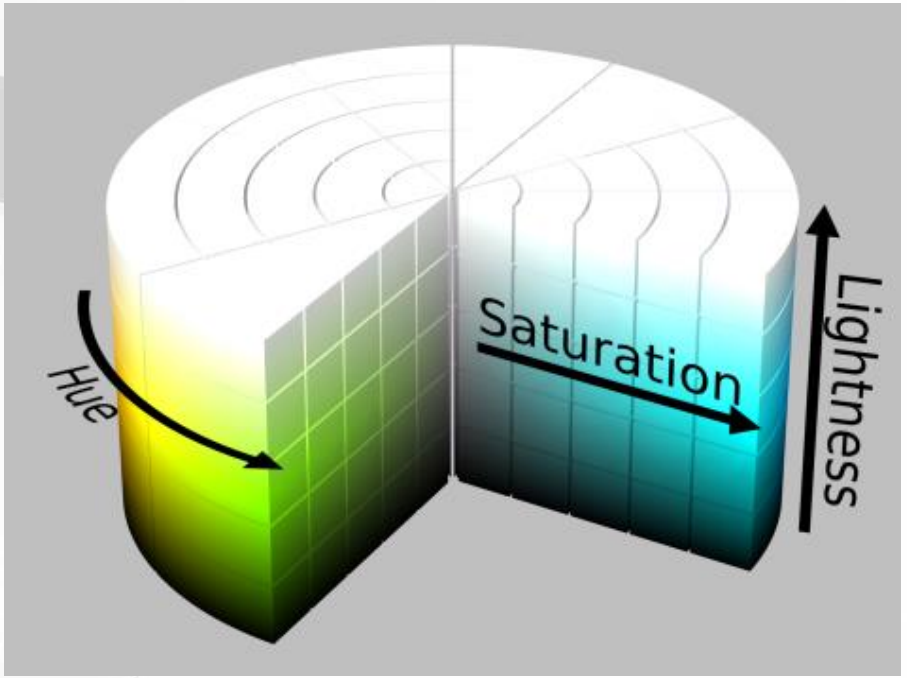
- Hue is a degree on the color wheel

  0 (or **360**) is red, **120** is green, **240** is blue

- Saturation is a percentage value

  100% is the full color

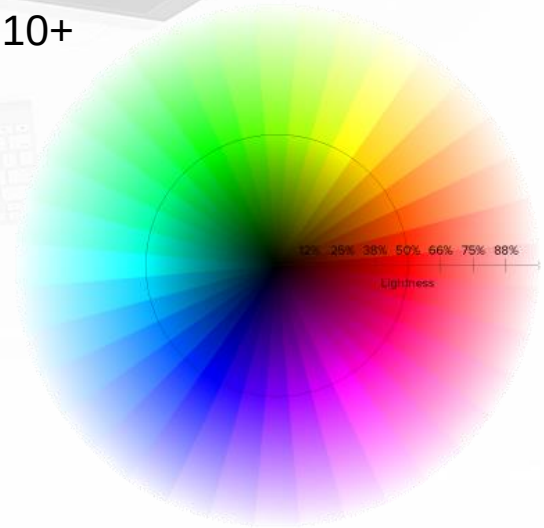
- Lightness is also a percentage
  - **0%** is dark (black)
  - **100%** is light (white)
  - **50%** is the average



HSLA Colors

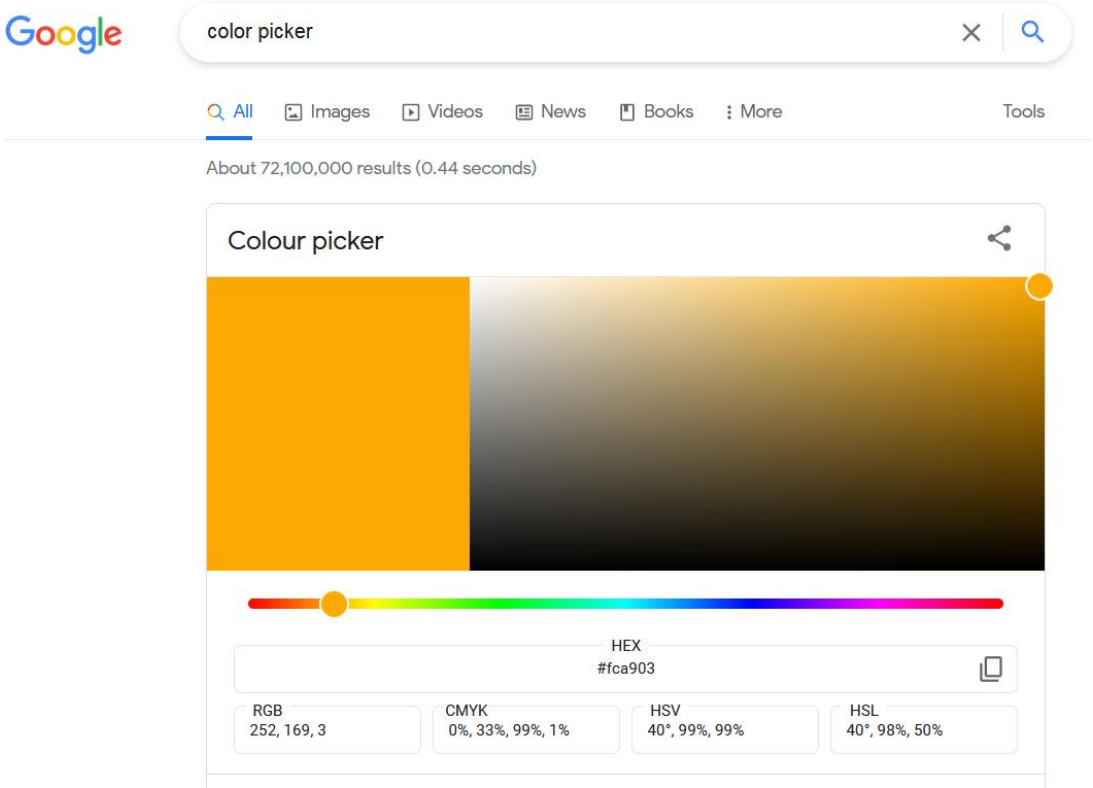
- HSLA allows a fourth value, which sets the Opacity (via the Alpha channel) of the element
- As RGBA is to RGB, HSLA is to HSL
- Supported in IE9+, Firefox 3+, Chrome, Safari, and in Opera 10+
- Example:

  hsla(0, 100%, 50%, 0.5)



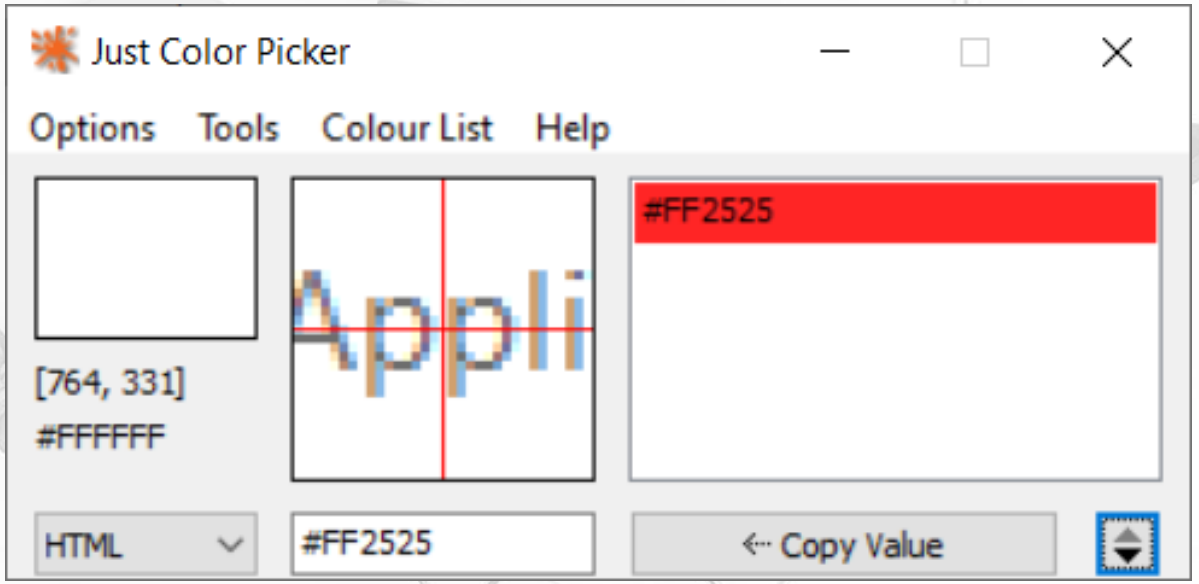
Color Picker Tools

Online Color Picker



Color Picker Tools

Just Color Picker



Default Browser Styles

Why Things Look Different on Different Browsers?

Default Browser Styles

- Browsers have **predefined** CSS styles

  Used when there is no CSS information or any other style information in the document

- **Caution:** default styles differ in browsers
  - E.g. margins, paddings, and font sizes differ most often
  - Usually developers **reset** them

```
* { margin: 0; padding: 0; }
body, h1, p, ul, li { margin: 0; padding: 0; }
```

CSS Cascade (Precedence)

There are browser, user and author stylesheets with “**normal**” and “**important**” declarations

- **Browser styles (least priority)**
- **Normal declarations in author stylesheets (external < in head < inline)**
- **Important declarations in author style sheets**
- Important user styles (max priority)
- Important declarations in user agent stylesheets

```
a { color: red !important; }
```

CSS Specificity

CSS specificity is used to determine the precedence (priority) of the CSS style declarations with the same origin

- **Simple calculation:**

  #id = 100, .class = 10, :pseudo = 10, [attr] = 10, tag = 1, * = 0

- **Same** number of **points**? **Order** matters!
- See also:
  - http://www.smashingmagazine.com/2007/07/27/css-specificity-things-you-should-know/
  - http://css.maxdesign.com.au/selectutorial/advanced_conflict.htm

CSS Specificity

Example

Selector	Thousands	Hundreds	Tens	Ones	Total specificity
`h1`	0	0	0	1	0001
`h1 + p::first-letter`	0	0	0	3	0003
`li > a[href*="en-US"] > .inline-warning`	0	0	2	2	0022
`#identifier`	0	1	0	0	0100
No selector, with a rule inside an element's `style` attribute	1	0	0	0	1000

What happens when CSS conflicts occur?

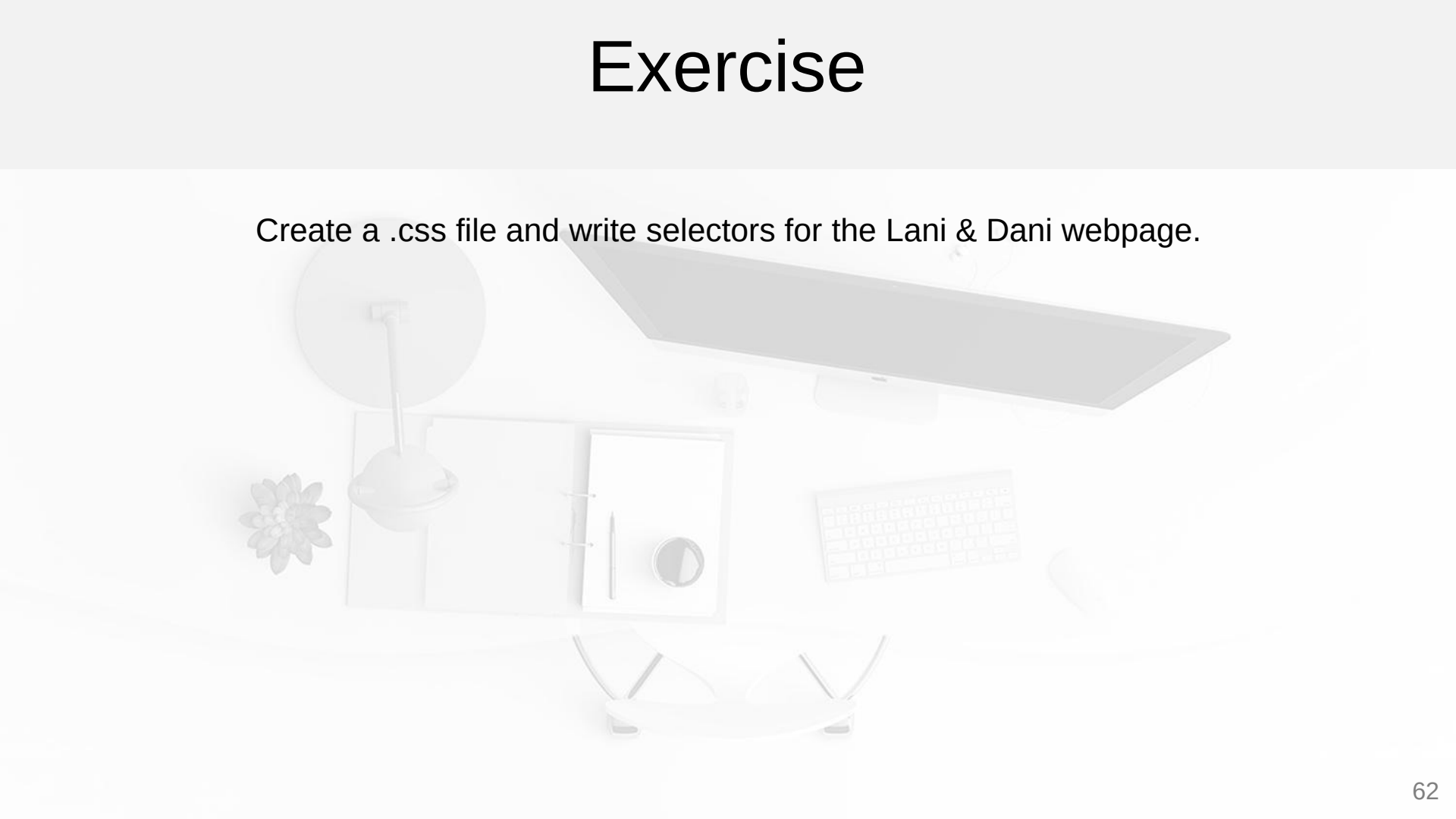
1. **Find all** declarations whose **selectors match** a particular element
2. **Sort** these declarations **by weight and origin (source order)**

   Note: !important
3. **Sort** the selectors **by specificity** (ID > Class = pseudo = attribute > tag > *)
4. **Sort by order** specified (the same tag)


Summary

Exercise

Create a .css file and write selectors for the Lani & Dani webpage.




Q&A