Școala
informală
de IT

Introduction in CSS

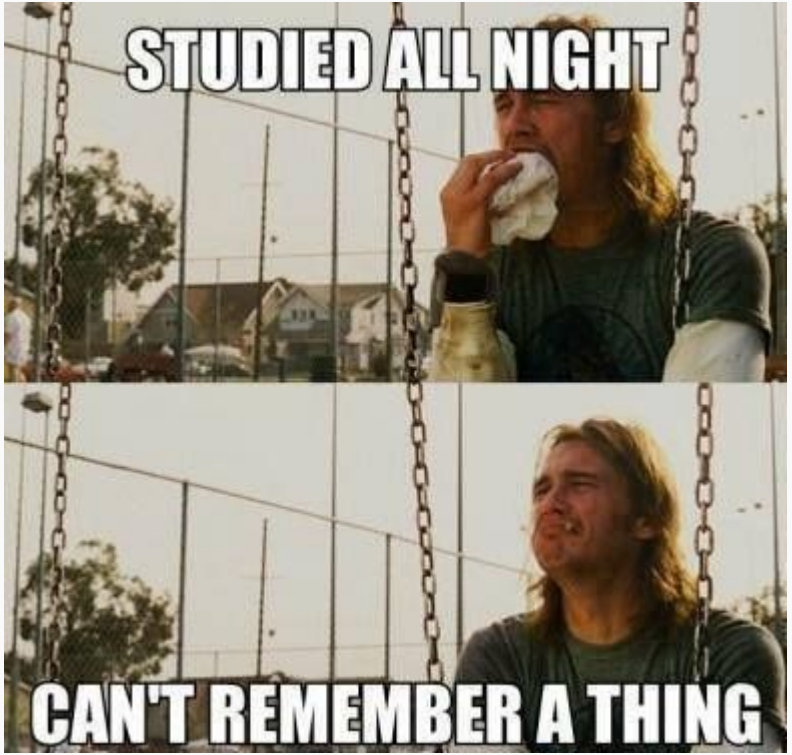
Agenda

- Recap
- CSS Overview
- Linking CSS and HTML
- CSS Syntax
- CSS Selectors
- Cascading Rules

Recap

Last session’s topics

- HTML definition and characteristics
- HTML Document Structure
- HTML Elements
- HTML Tables



CSS Overview

Web Page Building Blocks (again □)

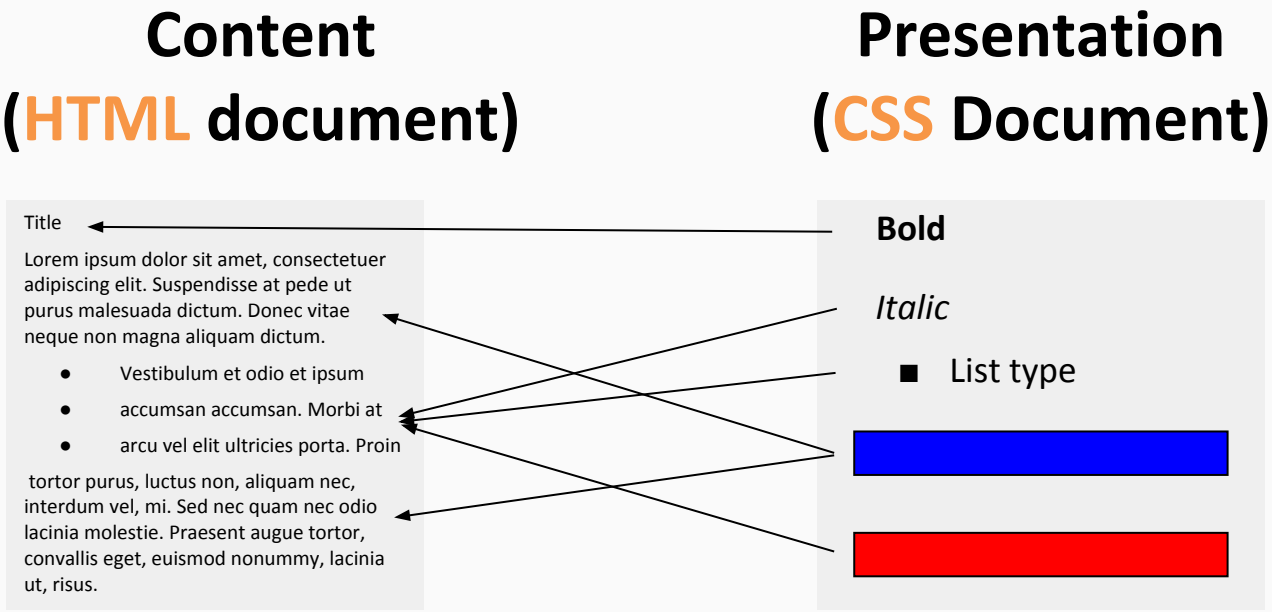
- HTML
  - describes and defines the structure and content of a webpage
- CSS
  - describes the appearance or presentation of content on a web page
- JavaScript
  - adds interactivity and other dynamic features to the web application

CSS Definition and Characteristics

- = a language used to define the visual appearance of web pages
- Allows web to separate content and structure of a website from the visual appearance
- Was created in 1997, but started gaining popularity in 2000
- Current standard: CSS 3
  - Implemented by most browsers
  - http://css3test.com/
  - https://caniuse.com/
- CSS can specify different styles for different media: on-screen, print, tablets, smartphones, projectors, and even by voice or Braille-based reader

CSS

- **Separating content from presentation**



What is Rendered

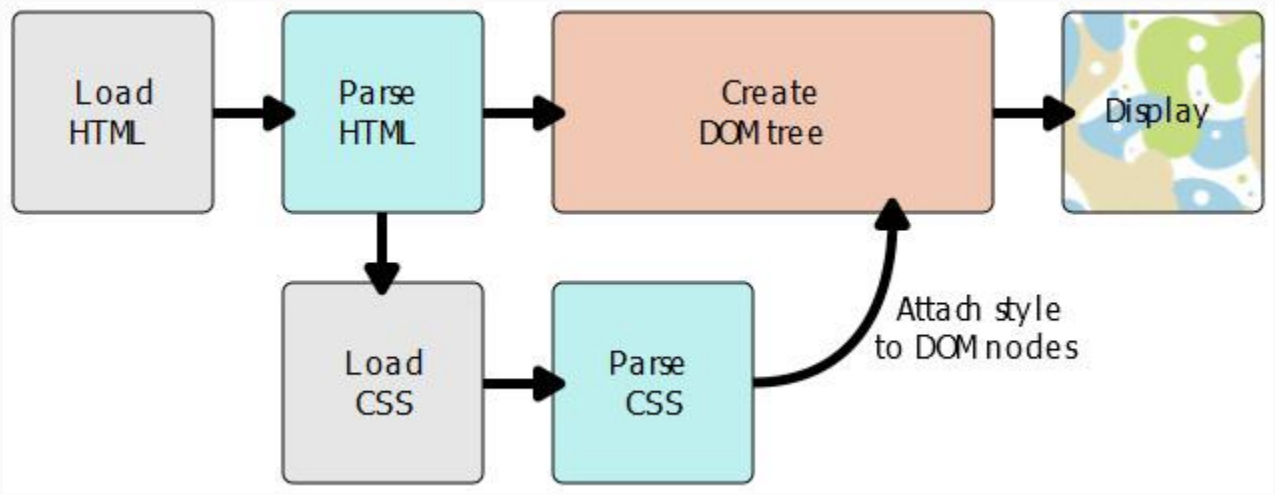
Title

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Suspendisse at pede ut purus malesuada dictum. Donec vitae neque non magna aliquam dictum.

- *Vestibulum et odio et ipsum*
- *accumsan accumsan. Morbi at*
- *arcu vel elit ultricies porta. Proin*

Tortor purus, luctus non, aliquam nec, interdum vel, mi. Sed nec quam nec odio lacinia molestie. Praesent augue tortor, convallis eget, euismod nonummy, lacinia ut, risus.

CSS and HTML


Load HTML
Parse HTML
Create DOM tree
Display
Load CSS
Parse CSS
Attach style to DOM nodes

Cascading Style Sheet

- **Style Sheet**
  - refers to the document itself
  - CSS files are text documents that can be edited with a variety of programs
- **Cascading**
  - The style sheet is intended to cascade through a series of style sheets, like a river over a waterfall
  - The water in the river hits all the rocks in the waterfall, but only the ones at the bottom affect exactly where the water will flow

Cascading

- Determine which style rules apply to an element using a priority scheme
  - 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

Cascading continued

- Some CSS styles are inherited and some not
  - Text and list-related properties are inherited: `color`, `font-size`, `font-family`, `line-height`, `text-align`, `list-style`, etc
  - Box-related and positioning styles are not inherited: `width`, `height`, `border`, `margin`, `padding`, `position`, `float`, etc
  - `<a>` elements do not inherit color and text-decoration

Linking CSS and HTML

CSS Syntax

- Simply put, web browsers apply CSS rules to affect how a document is displayed
- A CSS rule is made of:
  - A set of properties, which have values set to update how the HTML content is displayed
  - A selector, which selects the element(s) you want to apply the updated property values to

Linking HTML and CSS

- **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
    - Usually a file with `.css` extension
    - Linked via `<link rel="stylesheet" href=...>` tag or `@import` directive in embedded CSS block only

Inline Styles: Example

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

Embedded Styles

- Embedded in the HTML in the `<style>` tag:

```
<style>
    ...
</style>
```

  - The `<style>` tag is placed in the `<head>` section of the document
  - Used for document-specific styles

Embedded Styles: Example

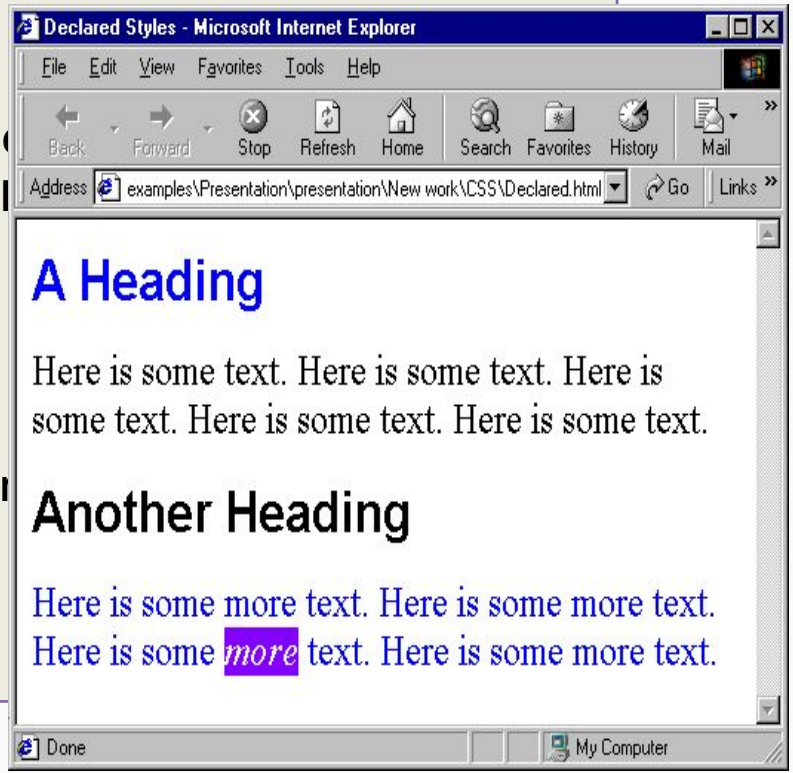
embedded-stylesheets.html

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

Embedded Styles: Example

embedded-stylesheets.html

```
<body>
  <h1 class="blue">A Heading</h1>
  <p>Here is some text. Here is some
  is some text. Here is some text. H
  text.</p>
  <h1>Another Heading</h1>
  <p class="blue">Here is some more
  Here is some more text.</p>
  <p class="blue">Here is some <em>m
  text. Here is some more text.</p>
</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 (see http://www.csszengarden.com/)
- `link` tag (with a `rel` attribute)
  - Specifies a relationship between current document and another

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

  - `link` elements should be in the `<head>`

External CSS Styles

styles.css

```
em {
     background-color: #8000FF;
     color: white;
}

h1 {
    font-family: Arial, sans-serif;
}

p {
    font-size: 18pt;
}

.blue {
    color: blue;
}
```

Default Browser Styles

- Browsers have default CSS styles (user-agent styles)
  - Used when there is no CSS information or any other style information in the document
- Caution: default styles differ from browser to browser
  - E.g. margins, paddings and font sizes differ most often and usually developers reset them

```
* { margin: 0; padding: 0; }
```

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

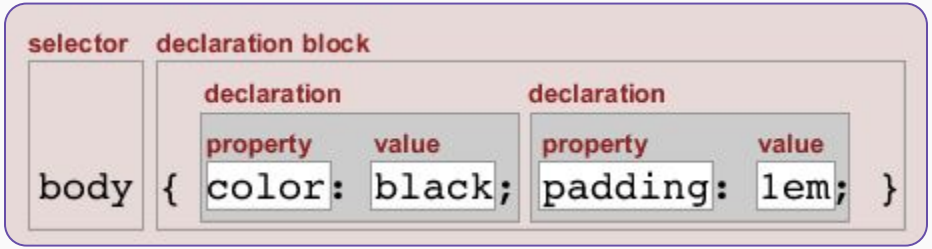
CSS Syntax

CSS Syntax

- Simply put, web browsers apply CSS rules to affect how a document is displayed
- A CSS rule is made of:
  - A set of properties, which have values set to update how the HTML content is displayed
  - A selector, which selects the element(s) you want to apply the updated property values to

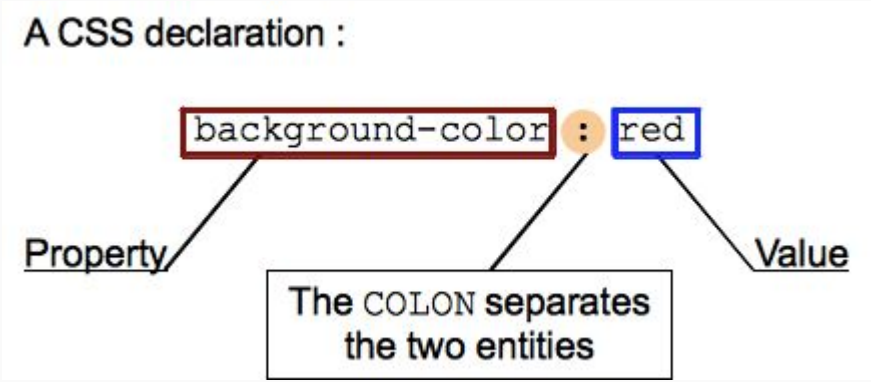
CSS Syntax in a different angle

- Stylesheets consist of rules
  - Declarations
    - Properties
    - Values
  - Selectors
- Selectors are separated by *commas*
- Declarations are separated by *semicolons*
- Properties and values are separated by *colons*



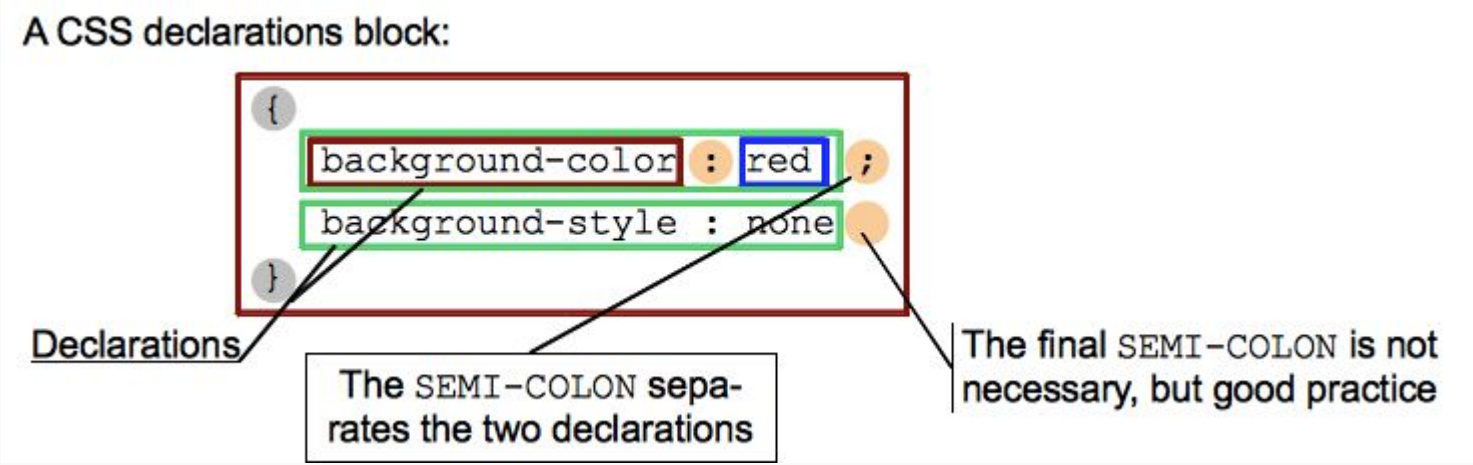

```
h1,h2,h3 {
  color: green;
  font-weight: bold;
}
```

CSS Declaration



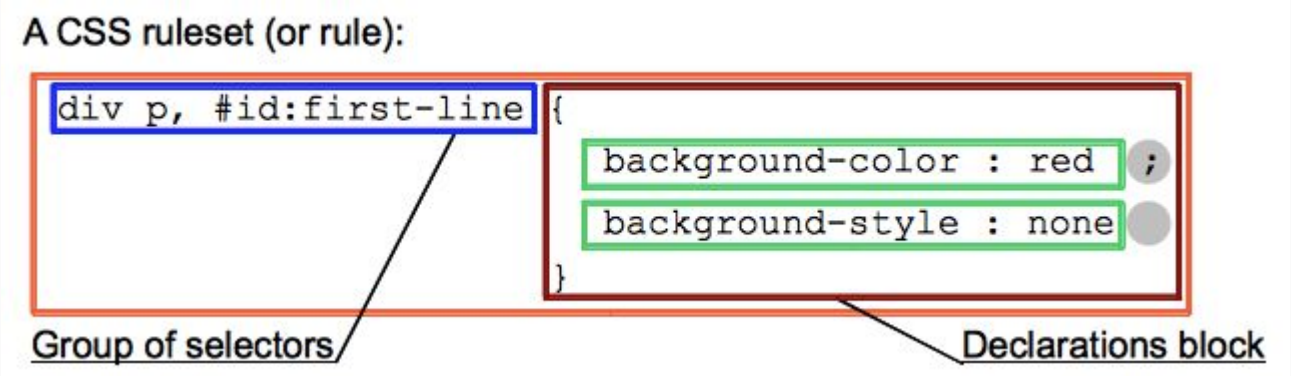
- If a property is unknown or if a value is not valid for a given property, the declaration is deemed invalid and is wholly ignored by the browser's CSS engine
- Use US spelling for properties
  - color, not colour

CSS Declaration Block



- **A CSS declaration block may be empty**

CSS Rules



- If a single basic selector in a chain or group is invalid, like when using an unknown pseudo-element or pseudo-class, the whole group of selectors is invalid and therefore the entire rule is invalid and ignored.

CSS Selectors

CSS Selectors

- Simple selectors
  - Based on element type, class or id
- Combinators
- Attribute selectors
  - Based on element's attribute or attribute values
- Pseudo-classes
  - Elements that exist in a certain state (hovered element, disabled checkbox)
- Pseudo-elements
  - Match parts of content that are in a certain position in relation to an element
- Multiple selectors
  - Apply same rules to elements selected by multiple selectors

Simple Selectors

- Type / Element selectors

```
h1 { font-family: verdana,sans-serif; }
```

- Class selectors
  - .className

```
.element_class { border: 1px solid red; }
```

- Id selectors
  - #id
  - An ID must be unique in the document!

```
#element_id { color: red; }
```

Simple Selectors Workshop

- Given the following HTML

```
...<div id="container">
    <p class="red">This is some text</p>
    <p>This is <span>more</span> text</p>
    <p>Link to <a href="http://www.scoalainformala.com">
        Scoala Informala</a></p>
    <a href="http://www.google.com">Google</a>
</div>...
```

a. Add a *background-color: gray;* style to the container
b. Style in red the first paragraph and the span
c. Style in blue the links

Combinators

- Universal selector (*performance!*)

```
* { font-family: verdana,sans-serif; }
```

- Descendent selector
  - space - select an element nested somewhere inside another element

```
div p { border: 1px solid red; }
```

- Child selector
  - > - select an element that is the immediate children of another element

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

Combinators (contd.)

- Descendent selector
  - + - select an element that is an immediate sibling of another element (i.e. right next to it, at the same level in the hierarchy)

```
div + p { color: blue; }
```

- Child selector
  - ~ - select any elements that are siblings of another element (i.e. at the same level in the hierarchy, but not necessarily right next to it)

```
div ~ p { border: 1px solid red; }
```

Combinators Workshop

- Given the following HTML

```
...<div id="container">
    <p class="red">This is some text</p>
    <p>This is <span>more</span> text</p>
    <p>Link to <a href="http://www.scoalainformala.com">
        Scoala Informala</a></p>
    <a href="http://www.google.com">Google</a>
</div>
<p>This is <span>some other</span> text</p>...
```

a. Style in blue the link to Scoala Informala, and in green the link to Google
b. Style in pink the "some other" span

Attribute Selectors

- [attr] selector
  - select all elements with the attribute `attr`, whatever its value

```
[id] { color: blue; }
```

- [attr=val] selector
  - select all elements with the attribute `attr`, and with value = `val`

```
[id="container"] { border: 1px solid red; }
```

- [attr~=val] selector
  - select all elements with the attribute `attr`, but only if the value `val` is one of a space-separated list of values contained in `attr`'s value

```
[class~="error"] { color: red; }
```

Substring Attribute Selectors

- [attr|=val]selector
  - select all elements with the attribute `attr` for which the value is exactly `val` or starts with `val-`

```
[id|="main"] { color: blue; } // main-article
```

- [attr^=val] selector
  - all elements with the attribute `attr` for which the value starts with `val`

```
[href^="https"] { color: blue; }
```

```
[class$="kg"] { color: green; }
```

- [attr$=val] selector
  - all elements with the attribute `attr` for which the value ends with `val`
- [attr*=val] selector
  - all elements with the attribute `attr` for which the value contains the string `val`

Pseudo Selectors

- Pseudo-selectors don't select actual elements, but rather certain parts of elements, or elements only in certain contexts
- Pseudo-classes
  - added on to the end of selectors to specify that you want to style the selected elements only when they are in certain state

```
selector:keyword { color: blue; }
```

- Pseudo-elements
  - added to the end of selectors to select a certain part of an element

```
selector::keyword { border: 1px solid red; }
```

Structural Pseudo-Classes

- :first-child - first element among a group of sibling elements
- :last-child - last element among a group of sibling elements
- :only-child - an element that has no siblings
  - How can we write this using :first-child and :last-child?
- :nth-child(an+b) - element whose numeric position in a series of siblings matches the pattern an+b
  - How to select every second child?
- :first-of-type - first element of its type among a group of sibling elements
- :nth-of-type(an+b) - element that has an+b-1 siblings of the same type before it, where n is positive or zero
- :nth-last-of-type(an+b) - same as nth-of-type, but it counts items backwards from the end, not the beginning

State-Related Pseudo-Classes

- :enabled - any enabled element: if it can be activated (e.g. selected, clicked on or accept text input) or accept focus
- :disabled - the opposite of enabled
- :checked - any radio button, checkbox or select option that is checked or toggled to an on state
- :link - links that have not yet been visited
- :visited - links that have been visited
- :hover - when the user designates an element with a pointing device, but does not necessarily activate it
- :active - an element (such as a button) that is being activated by the user
- For links, use the LVHA (LoVe-HAte) order for pseudo-classes, otherwise they will be overwritten!

More Pseudo-Classes

- **:not(*X*) - matches elements that are not represented by the argument. X must not contain another negation selector**
  - **This selector only applies to one element; you cannot use it to exclude all ancestors.**
  - For instance, **`body :not(table) a`** will still apply to links inside of a table, since <tr> will match with the :not() part of the selector.
- **:target - a unique element (the target element) with an id matching the URL's fragment**
  - https://developer.mozilla.org/en-US/docs/Web/CSS/:target

Pseudo-Elements

- ::after - creates a pseudo-element that is the last child of the selected element
- ::before - creates a pseudo-element that is the last child of the selected element
- ::first-letter - first letter of the first line of a block-level element, but only when not preceded by other content (such as images or inline tables)
- ::first-line - first line of a block-level element
- ::selection - the portion of a document that has been highlighted by the user (such as with the mouse)
  - can't apply background-image

```
HTML:
<q>Some quotes</q>, he said, <q>are better than none</q>.

CSS:
q::before {
  content: "«";
  color: blue;
}
q::after {
  content: "»";
  color: red;
}
```

Multiple Selectors

- You can also select multiple types of elements and apply a single rule set to all of them
  - Include multiple selectors separated by commas

```
div#container, span:only-child, h1  { color: blue; }
```

Cascading Rules

Cascading Rules

- **Three factors that determine which CSS style wins:**
  1. **Importance**
  2. **Specificity**
  3. **Source Order**

(in the order of their “weight”)

Importance

- A certain rule will always win

```
div { color: blue !important; }

div#container { color: red; }
```

- The only way to override this !important declaration would be to include another !important declaration of the same specificity, later in the source order
- DO NOT USE !important !!!

Specificity

- = a measure of how specific a selector is
- 4 different values:
  1. Thousands
     - if the matching selector is inside a <style> element or the declaration is inside a style attribute
  2. Hundreds
     - IDs
  3. Tens
     - Classes, attributes or pseudo-classes
  4. Ones
     - Elements or pseudo-elements
- Universal selector (*), combinators (+, >, ~, ' ') and negation pseudo-class (:not) have no effect on specificity.

Specificity Workshop

Selector	Thousands	Hundreds	Tens	Ones	Total
div					
#container					
p + a::first-letter					
li > a[href=*"en-US"] > .red					
#container div > p > a:hover					
#container div > p > a:hover inside a <style> element					

Specificity Workshop

Selector	Thousands	Hundreds	Tens	Ones	Total
div	0	0	0	1	0001
#container	0	1	0	0	0100
p + a::first-letter	0	0	0	3	0003
li > a[href=*"en-US"] > .red	0	0	2	2	0022
#container div > p > a:hover	0	1	1	3	0113
#container div > p > a:hover inside a <style> element	1	1	1	3	1113

Source Order

- later rules will win over earlier rules
- Is a factor when multiple competing selectors have the same importance *and* specificity

Readings & Tutorials

- http://www.learn-html.org/en/Styles
- http://benhowdle.im/cssselectors/
- https://www.smashingmagazine.com/2016/05/an-ultimate-guide-to-css-pseudo-classes-and-pseudo-elements/
- https://specificity.keegan.st/
- https://www.codecademy.com/en/courses/learn-html-css/