MIDTERMS NOTES

**Selector** – target you want to format

- Whatever you indicate is a subject

- Chain or 1 more sequence of simple selector separated by combinators

- Always start with a type selector or universal selector

-

\* means everything

div > \* - anything that is a child of div

\*.\_\_ - everything with that(value)

**Type selector** – p, h1, etc.

**ID selector** -> #

Nav # main

**Class Selector**

-ui.top-ten

-.normal

“ ~= ” matches the word, “ /= ” anything begins with … /- , “ ^= ” anything start with https, “$=”- don’t know, “ \*=” anything who’s value ….

**Attribute Selector** – particular/ specific/ direct match

Img[alt]

Nav[id = main]

**Combinators** used to impose additional matching constrains

- Descendants / not necessarily a direct descendant

- Child combinator(>)

- Sibling combinator

o Adjacent(+)

o General (~)

Selector group - “ , ” union of all elements

Pseudo elements/classes

- Dynamic pseudo class

o Link pseudo class - :link, :visited

o User action pseudo class - :hover, :active, :focus

- Target pseudo class - :target

- Language pseudo class - :lang()

- UI element State Pseudo class - : enable, :disable. :checked, :indeterminate

- Structural pseudo class - :root, :first-child, :last-child, :only-child, :nth-child(), :nth-last-child(), :first-of-type, :last-of-type, :only-of-type, :nth-of-type(), :last-of-type(), :empty

- Pseudo element - : first-letter, :first-line, :before, :after

Content: ‘chapter’ counter

Counter – increment: chapter 1

Counter – reset : topic 0

* CSS Rule Precedence
  + HTML element may be subject of selectors of multiple style rule.
    - Whenrules target different properties: effects CASCADE (they combine)
    - When styles involve same properties they CONFLICT and must be resolved.
    - Resolution
      * By ORIGIN and IMPORTANCE
        + Origin - author / user-agent
        + Importance - priority
      * By SPECIFICITY - that is calculated based on three sequence**.** Specificity only applies when the same element is targeted by multiple declarations.

1. Type selectors (e.g., h1) and pseudo-elements (e.g., ::before).
2. Class selectors (e.g., .example), attributes selectors (e.g., [type="radio"]) and pseudo-classes (e.g., :hover).
3. ID selectors (e.g., #example).t

* When multiple declarations have equal specificity, the **last declaration found in the CSS is applied to the element**

SEE LINK: <https://specificity.keegan.st/>

* CSS DECLARATIONS
  + Properties
    - Shorthand properties
      * Allows authors to specify the value of several properties.
      * Single property (eg. border,bg, font, margin, border-left/ right)
        + background-color:gray;
        + font-size:22px;
  + Vendor specific extension (Also known as **vendor prefixes**)
    - Used by browser vendors as prefix for names of experimental / non- standard CSS properties
      * Eg. -webkit, -moz-,-o-,-ms-
      * Lately vendors are moving away from vendor prefixes in favor of user controlled flags
        + Eg. -webkit-border-radius:20px;
  + Custom properties (Also known as **CSS variables**)
    - Property names, prefix with -- representing a value that can be reused throughout a document.
      * Eg. element { --main-bg-color: brown;}
      * element { background-color: var(--main-bg-color);}

* CSS Values
  + Value Processing
    - Declared, Cascaded, Specified (When you declare), Computed(Relative value), used and read on how values are being processed.
  + Value Types
    - Keywords : Initial, Inherit, Unset
      * Eg. h1{color:blue;}
      * div {color: red;}
      * h1{color: inherit;}
      * h1{color: unset;} - RESETS THE COLOR
    - Property specificity
    - Numbers
    - Dimension
      * Length, angle, durations.
      * Length units
        + font relative : em, ex, rem
        + Viewpoint percentage
        + Absolute, cm, mm
      * Duration
        + s,ms - used in animation / transitions
      * Frequency units
        + Hz/KHz (Initially introduced in CSS2 now obsolete)
      * Resolution units
        + Dppx,apr,dpcm
      * Angle units
        + Deg, rad, grad - used in some gradient and transform functions.
      * Percentage
        + Number with a ‘%’ suffix eg. 5%
        + Used to calculate a percentage of some value
      * URL/ URI
        + url()function with absolute or relative. URL parameter
      * Colors
        + Color keywords: red
        + RGB hexadecimal notations : ff0000
        + RGB function (255,0,0) / rgb(100%, 0%,0%)
        + HSL functions(255,0,0) - hue Saturation and lightness
        + SEE:<http://www.colour-affects.co.uk/psychological-properties-of-colours>
      * Strings
        + Delimited by (‘) or (“)
      * Functions
      * Miscellaneous types

* CSS PREPROCESSORS / FRAMEWORK & POLYFILLS
  + CSS preprocessors
    - Generate css using a custom language syntax that includes features not included in CSS
      * SASS - “allows you to use nested CSS/ can apply css without using brackets”
      * SCSS {less} see <https://sass-lang.com/>
    - Mixins : information you can reuse.

Eg. mixin abc{

Font: italic sans-serif; border : 2px dashed;

}

Call the mixin in a CSS:

@include abc(1px, gray)

* + CSS frameworks
    - Provides pre defines CSS designs functions that can be reused / extended eg. bootstrap, Foundation

* JAVA SCRIPT
* Mag type ka naman.

JavaScript Execution Environment

* Core JavaScript(EcmaScript) Language and API
* (Traditional) Browser Object Model(BOM) API(non-standardize)
  + Window, Navigator, Screen, Location, History
* Document Object Model(DOM) API
  + Node, Document, Element, TExt, Attr, DocumentType, Comment, etc.
  + Event, EventTarget, EventListener, etc.
  + CSSStyleSheet, CSSRule, etc.
* Miscellaneous JavaScript Web APIs
  + Geolocation, IndexedDB, Local Storage, Push Notifications, Service Workers, Web Sockets, Web Workers, XMLHttpRequest(AJAX), etc.

Document Object Model(DOM)

* API that allows access to the HTML document from within scripts associated with the web page.
  + A parsed HTML document is represented by a DOM tree, Which contains nodes representing *elements, element attributes, textual content*, and other HTML document components(e.g., DOCTYPE, comments, etc.)
* Accessed via the ***document***object property of the global ***Window*** object.
* Functionalities:
  + Retrieve references to nodes(or node collections) in the DOM
    - getElementById()
    - getElementByTagName(), getElementByClassName(), getElementByName()
    - querySelector(), querySelectorAll()
  + Traverse the DOM tree(from a given node)
    - parentNode, parentElement
    - childNode, children
    - firstChild, lastChild, nextSibling, previousSibling
    - firstElementChild, lastElementChild, nextElementSibling, previousElementSibling
  + constructor/copy nodes
    - createElement(), createTextNode(), createAttribute(), etc.
    - cloneNode(), importNode()
  + Manipulate the DOM tree
    - appendChild(), insertBefore(), replaceChild(), removeChild()
    - adoptNode()
  + Miscellaneous attributes and methods
    - noteType, nodeName, NodeValue
    - Attributes, id, tagName, className, classList
    - ownerDocument, documentElement, hasChildNodes()
    - normalize()
  + Manipulate attributes (for ***element*** nodes)
    - setAttribute(), getAttribute(), removeAttribute()
    - setAttributeNode(), getAttributeNode(), removeAttributeNode()
  + Event handling
    - addEventListener(), removeEventListener(). dispatchEvent()
  + Style handling
    - styleSheets, ownerRule, cssRules, insertRule(), deleteRule(), getComputedStyle()