CSS PART 2

PAGE AND VISIBILITY

AGENDA

- Page display and visibility rules
- The Box Model
- Styling links and lists
- Some different ways to specify CSS selectors

PAGE DISPLAY AND VISIBILITY RULES

HTML ELEMENTS DISPLAY AND LAYOUT

- Every element is treated as a *box* (discuss more when we learn the **box model** concept)
- Display of one element will affect the layout of neighboring elements
- In terms of page layout, you need to consider whether these boxes are laid out
  - Next to each other
  - One after other and in which order
  - Nested

LAYOUT AND DISPLAY– KEEP IN MIND THE FOLLOWING COMMON VALUES

- display: inline –
  - These elements are placed side by side (horizontal separation)
  - They will take up only the amount of width and height required to contain their content
  - You cannot control width and height
  - E.g. -
- display: block –
  - Causes a line break (takes up entire width and only the height required to display the content)
  - Elements will be placed one after another (vertically)
  - Can specify rules for width and height attributes to control layout
  - E.g. - and <p>

USING BEST VALUES OF INLINE AND BLOCK

- The inline-block value of css property, display
  - Functionally same as inline display (elements next to each other), however
  - You can specify width and height

- If you want to make the browser ignore element(s)
  - Set it to display *none*

DISPLAY PROPERTY FOR CSS RULES

- display: inline-block (as example. You can also use other values like block or none etc. try it out)
  - Note that you have to test this out to figure out what you want and what works best for your page layout
  - You have to keep in mind layout of one element can at times effect neighboring elements

OTHER LAYOUT PROPERTIES – FLOAT AND CLEAR

- float: –
  - Reposition layout of elements to left or right
  - Note if you do it to a group of elements they will not overlap
  - Available Values: left or right
- clear: –
  - It keeps floating elements away
  - if you don't want the element in question to be affected by neighboring elements that are floating
  - Available Values: left [tell browser there should be no floating elements to my left], right, both

PROPERTY OF ELEMENT - OVERFLOW

- Scenario – you set height and width of element, but content does not fit in the specified layout
- Use property **overflow**
  - Defines how users can access such content
- Values –
  - visible: might result in overflowing text to overlap each other
  - hidden: hides any content that overflows from the specified box
  - scroll: adds vertical and horizontal scrollbars
  - auto: adds scrollbars whenever the need arises

OTHER DISPLAY PROPERTIES

- Table
  - Table like layout without using the <table> html element
  - E.g. – display:table used in conjunction with display:table-cell for elements/columns
  - Try it out
- Grid
  - display:grid; followed by grid layout specs using grid: ‘menu footer footer … ‘
- Flexbox
  - display: flex; for a flexible layout

PROPERTY FOR ELEMENT - VISIBILITY

- Visibility property will specify whether an element is visible on the page
- Values include:
  - visible
  - hidden
  - collapse (applicable to table elements)
- **IMPORTANT DIFFERENCE**
  - display: none makes the browser treat the element as if it was never there
  - with visibility: none, there will a blank space in the webpage where the element should have been. You won't be able to see it.

THE BOX MODEL

A helpful tool to size and place your elements

PROPERTY - BORDERS

- Border –
  - You can put a border around any element
  - Border properties have *style, width, and color*
  - Specifying style is mandatory, the other two are optional
  - E.g. – border: solid 20px #FF22FF;
- border-style values:
  - none , dotted, dashed, solid, double, groove, ridge, inset, outset, hidden.
- border-width values:
  - Thin, medium or large; or specify custom using unit of pixel (px)
- border-color values: like you specify text color

IMPORTANT CODING NOTE 1 - SHORTHANDS

- Note that fact in the previous example we wrote:
  - border: value1 value2 value3;
  - Alternatively you can write this as
    - border-style: solid;
      border-width: 20px;
      border-color: ….;
  - If you combine multiple property subtype you can separate them by whitespaces, and write in a single line, BUT, you need to know the order of specification..

IMPORTANT CODING NOTE 11 - SHORTHANDS

- for border-width note the following
- If you do this:
  - border-width: 5px; // this is going to put a border of 5px on all four sides (top, right, bottom, left)
- If you do this:
  - border-width: 2px 5px; // will use 2 px border on top and btm and 5px on right and left
- If you give three values:
  - border-width: 2px 3px 5px // first one is top, next is right and left, last is bottom
- If you give 4 values:
  - border-width: 1px 2px 3px 4px // **top→right→bottom→left**
  - Remember this order – its clockwise starting from top. This will apply to property of margin & padding as well.

PROPERTY - MARGIN

- Margin add additional space **outside** your borders.
- It's the separation between one html element and another i.e. neighboring elements
- Positive margin value indicative of
  - move elements to the right or down
- Negative margin value indicative of
  - move elements to left and upward [move closer to neighbor]

PROPERTY - PADDING

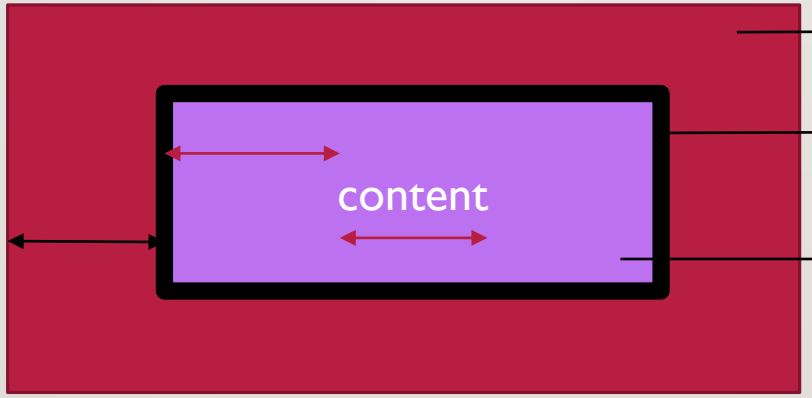
- Padding adds additional space **between** the element's contents and its borders

- Positive value
  - border moves away from element
- Negative value
  - border moves closer

OTHER NOTES ON MARGIN AND PADDING

- They don't take any color values
- They can be defined using shorthand notations of 1 to 4 values as done for borders

VISUALIZE BORDER, MARGIN AND PADDING



Margin is red

The black color is border

Purple is padding.
Padding is space between
Start of element and start
Of contents

Total width of an element = margin+border+padding+content width
Remember that width and height are additive across different properties

ALIGNING ELEMENTS – CENTERING ELEMENTS

- Text aligning is easy. If you want to center text or justify text alignment
- But to center elements you should remember this:
  - margin: 0 auto;
  - works only if
    - display: block
    - no float property used
    - **element does not have fixed or absolute positions**
    - elements width is not auto

BOX SIZING

- Box-sizing will not require you to calculate everything

- Options used:
  - content-box: default additive
  - border-box: width (considers content, padding, and border)

SPECIFYING WIDTH AND HEIGHT – UNITS OF MEASUREMENTS

- Absolute : set to a specific number with a unit of measurement
  - px, cm, mm, pt, and more

- Fluid – size relative to surrounding elements
  - %, vw, vh
  - em (for font sizes): 1 em refers to current size. 0.75 em – 75% of current size
  - rem (for fonts): 1 rem is current size of root element

REMEMBER!!

- Design your layout before you start coding
- Use the box model to sketch your design
  - Incorporate the idea of margin, then border, then padding, then content for any element
- Margin property must be taken into considerations while designing pages
- Practice, practice, and practice
  - The more you tinker around with these elements and properties, the more your confidence will grow.

STYLING RULES FOR LINKS AND LISTS

STYLING LINKS

- You can use all the previously learned rules for normal texts and display properties and apply it to style links
- In addition, you can use a property specific to links called *text-decoration*
- A use-case scenario to consider:
  - when you put a text-based link in HTML, it has a underline.
  - Many designers tent to remove this underline – making the hyperlink look like a **button**
  - **Don't do this. If you need a button use a <button> and not an anchor tag, <a>**
    - Html is about semantics!
- So when you style links, make sure it still resembles a link and reads out like a link

COLOR OF LINKS -- STATES

- While coding links, you will notice some links are different colors
  - Some show up as blue, some as purple
  - These states can be styled by knowing the types of link states
- a:link – a normal and unvisited link
- a:visited – a link that has been visited
- a:hover – activates on hover the mouse over the link [what about touchscreen devices?]
- a:focus – activates using the keyboard tabs
- a:active – when a link is being clicked [only used meaningfully designing nav bars]

STYLING LINKS - PRECEDENCE OF RULES

- a:hover MUST be written AFTER a:link

- a:visited and a:active MUST be written AFTER a:hover

STYLING LISTS

- Again the same set of properties for styling texts and display apply to lists as well.
- But styling specific to lists involves things like changing the type of symbol used to represent ordered and unordered lists.
  - For example, the bullets on this slide are circle, and we want to change it to a box
  - That is an example of styling specific to lists

STYLING LISTS - AVAILABLE PROPERTIES:

- list-style-type
  - some values for <ol>: lower-roman, upper-roman, upper-alpha, Hebrew, and many more
  - Some values for <ul>: circles, discs, squares etc.
- list-style-image
  - use your own custom images as a marker
  - E.g. – list-style-image: url('path/to/your/image.png')
- list-style-position
- list-style

TOOLS TO HELP WITH GENERATING CSS OR WEB DEV TOOLS

- http://css3generator.com/

- https://chrispederick.com/work/web-developer/

ADVANCED SELECTORS

Some more ways for specifying CSS selectors

CSS SELECTOR SPECIFIERS BASED ON DOM

- CSS selectors specification can follow the DOM. Knowing the DOM is even more important otherwise you cannot engage with these selectors
- Descendant selectors
  - E.g. – nav a { } – select the anchor tags within nav tag
- Child selectors (strict selection based on DOM)
  - e.g. – nav > a { } – selected anchor elements must be a direct child of the nav tag itself
- Adjacent siblings
  - e.g. – h1+ol – elements are on the same level of the DOM and follow each other

CSS SELECTOR SPECIFIERS BASED ON ID AND CLASS

- Example for id
  - #id { } – identifies a single element in the DOM
  - an id should be unique across the DOM

- Example for class
  - .class { } – identifies a group of elements, for example – logo images
  - class names can be re-used

NARROW YOUR SELECTOR SCOPE

- Select all paragraphs which are specifier in class “main”
  - p.main { }

- Select all images that belong to class “special” and are present in the header element
  - header img.special { }

EXPAND YOUR SELECTOR SCOPE

- Apply styling rules to multiple elements
  - header, footer { .. }
  - h1,h2, .myClass {…}
- Universal selector : (select everything on page – not suggested unless used for debugging)
  - * { }
- Attribute Selectors : (select a specific set of attributes from a tag)
  - a[href='index.html']
- You also have Pseudo-classes and Pseudo elements – will discuss later

ATTRIBUTE SELECTORS

- Example Scenarios where attribute selectors can be useful:
  - find images on the page that uses a specific extensions
  - find all images on the page that have empty alt text
  - find all links that redirect to .gov sites etc.
- Operators used with attribute selectors;
  - ^ : match the beginning exactly
    - a[href^='https://oakland']
  - $: match the end exactly
    - img[src$='.jpeg']
  - * : wildcard
    - a[href*='valve']