More Properties

Box Model

- **border**: any element can have a border around it
  helps you visualize how much space an element takes up
  specifies style, width, & color
  - style MUST be specified; width & color have fall-back default values

  - border-style values: none, dotted, dashed, solid, double, groove, ridge, inset, outset, hidden
  - border-width values: pixels, thin, medium, large
  - border-color values: name/keyword, rgb, hex, transparent

  → (eg. border: solid 1px #CC00AA;):

- specifying border-width for individual sides (also works for margin & padding)

  → (eg. border-width: 3px;): 3px on all 4 sides

  → (eg. border-width: 3 px 10 px;): 3 px on top/bottom
  10 px on right/left

  → (eg. border-width: 3px 10 px 20 px;): 3 px on top
  10 px on right/left
  20 px on bottom

  → (eg. border-width: 3px 10 px 20 px 1 px;): 3 px on top
  10 px on right
  20 px on bottom
  1 px on left

- **margin** = additional space outside border
  between element & element's neighbor
  transparent → takes on parent's color

  - positive margin = element moves right/down
  away from neighbor
  - negative margin = element moves left/up
  closer to neighbor

- **padding** = additional space between the element and its border
  transparent → takes on element's color

- positive padding = border moves outward from element - CSS doesn't support negative padding

- height & width are additive
- margin + border + padding + width/height = actual width/height

- horizontally center an element w/ margin

  - format: margin: 0 auto;

  - criteria: element must display: block;
    element must not float
    element must not have a fixed or absolute position
    element must have a width that isn't auto

- box-sizing: takes away the math

  - values: content-box
    border-box

  - content-box: default, additive

  - border box considers: content
    padding
    border

- measurements: absolute
  fluid

- absolute = set to a specific size

  - eg: px
    mm
    cm
    pt

- fluid = sets size relative to surrounding elements
  for best viewing

  - eg: %
    vw
    vh
    em] for font
    rem]

Styling Links & Lists

- text-decoration: to style anchor links

→ (eg. text-decoration: none;): gets rid of underline that appears on links

- style a link & make it still look like a link
- if it looks like a button → be semantic & use <button>

- link states

  - a:link — a normal, unvisited link
    a:visited — has been visited
    a:hover — activated by mouse
    a:focus — activated by keyboard
    a:active — is being clicked

  (hover & focus: n/a to touchscreens)

  must come first ↓ precedence order ↓ must come after

  - precedence rules: a:hover must come after a:link and a:visited
    a:active must come after a:hover

- styling lists beyond font, margin, etc: list-style-type
  list-style-image
  list-style-position
  → list-style

- list-style-type: to style list marker

  - defaults: ol - numbers
    ul - bullet point

  - ol values: lower-roman
    upper-roman
    decimal
    decimal-leading-zero
    upper-alpha
    lower-alpha

  - ul values: circle
    disc
    square

- list-style-image: use a custom image > traditional marker

  - can include multiple values as alternatives/back-ups

  → (eg. list-style-image: square url("icon.gif");): square first
  picture as back-up

- link-style-position: where to place list marker

  - values: inside
    outside

- search & make use of 'developer tools'
- helpful developer tools: chrispederick.com/work/web-developer
  css3generator.com

Advanced Selectors

- CSS selectors that follow the DOM: descendant selectors
  child selectors
  general sibling selectors
  adjacent sibling selectors

- **descendant selector**: selects all descendants of a specified element, regardless of position in DOM tree

  - format: parent element descendant that will be styled {...}

    → (eg. nav a {...}): style all the anchor links inside a nav tag

- **child selector**: more constraining than ^
  selects all children of a specified element, no intermediate tags, must be a direct child

  - format: parent element > child element that will be styled {...}

    → (eg. nav > a {...}): style all anchor links that are direct children of a nav tag

- **general sibling selector**: selects second element when it comes after the first & they have the same parent

  - format: element ~ element that will be styled {...}

    → (eg. div ~ p {...}): style all p tags that are siblings of div tags

- **adjacent sibling selector**: more constraining than ^
  selects second element when it comes immediately after the first & they share same parent

  - format: element + element that will be styled {...}

    → (eg. div + p {...}): style all p elements that directly follow div tags

- id vs. class selectors:

  - #id {...}
    unique id / identify a single element
    eg. to visually signify the current page in a nav bar

  - .class {...}
    can be reused / identify a single element w/in a group/class of items
    eg. format many, but not all, images the same way (as thumbnails)

- narrowing the scope

  → (eg. p.main {...}): paragraphs w/ class = "main"

  → (eg. header img.special {...}): paragraphs inside header w/ class = "special"

- expanding the scope / combining elements / multiple selectors to be styled

  - format: element, element, element, element, ... {...}

  - multiple rules for the same selector → builds on top of previous rules
    same property → follows precedence & most recent
    unless a rule has !important

- **universal selector**: styles every element on the page

  - format: * {...}

attribute selector: styles based on attributes inside the tags

- format: type selector [attribute name = 'selector'] {...}

  → (eg. a[href='info.html'] {...}

- eg. all images that use gif files
  all images w/ empty alt text
  all links that are .gov

- operators: ^ match the beginning exactly
  $ match the end exactly
  * match anywhere exactly

- format: type selector [attribute name operator = 'selector'] {...}

  → (eg. a[href^='http://umich'] {...}): style all links that start w/ http://umich

  → (eg. img[src$='.png'] {...}): style all png images

  → (eg. a[href*='umich']: style all links including umich

Shorthand Rules

- will switching between background & background-color make a difference? → NO!
- background inputs: background-color
  background-image
  background-repeat
  background-position

  → (eg. background: #000 url("imgs/ocean.jpg") no-repeat fixed center;)

[Q. id & class selectors are part of the DOM]

Thinking Beyond Selectors

Browser Capabilities

- design for consistent appearance!
- default style sheet: easiest way to handle/eliminate browser differences
  css resets in an external style sheet

- handle unsupported CSS3 properties w/ browser prefixes
  browser prefixes (aka vendor prefixes): -webkit- android, chrome, ios, safari
  -moz- firefox
  -ms- internet explorer
  -o- opera

- Check which properties need prefixes: http://caniuse.com
- Often unsupported properties: column-count
  gradient

- automated prefix adding: editor add-ons
  outside programs that dynamically add appropriate prefix based on browser

Background Images & Opacity

- link images in imgs folder from css file in css folder: ../ "go up" one folder level

  ⟶ (eg. background-image: url('../imgs/ocean.jpg');)

- opacity = specifies how transparent an element is

  - range: 0-1

  - 0 = invisible
    1 = opaque/solid

Designing for Accessibility

- content of page should be in HTML → don't add content via colors, images, etc
- follow the POUR guidelines: Perceivable
  Operable
  Understandable
  Robust

- be perceivable: provide alt text for images
  provide captions & transcripts for video & audio
  use correct semantic markup
  use good color contrast

- be operable: all functionality available through the keyboard
  provide controls for multimedia
  don't flash content / don't cause seizures

help users navigate, find content, and determine where they are

- be understandable: economical & plain use of language
  text supplemented w/ images, videos, etc where appropriate → use good Universal Design
  navigation & info structure are discernable & consistent
  make pages operate in predictable ways
  help users avoid & correct mistakes

- be robust: is your site functional across various technologies?
  syntax errors might not affect visuals but may hamper assistive tech & accessibility tools
  adhere to W3C standards → ensure future compatibility w/ new browsers
  validate code

- utilize accessibility tools!!
- cool design/style should not be at the cost of accessibility !!