

# Welcome

## Syllabus

- **CSS** = Cascading Style Sheets  
a method of styling your HTML documents w/ various colors, fonts, layouts, and spacing
- course focus:
  - rules & syntax
  - simple styling
    - fonts
    - colors
    - borders
  - more styling
    - bg images
    - opacity
  - positioning
  - pseudo-classes
    - styling things that aren't even there

# Getting Started with Coding

## Cascading Style Sheets

- same html may look different on different browsers: some tags are /n't supported  
browsers may have different default styles

- `html style attributes` → violated separation of content & style → `CSS`

- **CSS**: defined generic rules that can apply to multiple elements

```
-format: selector {
    property: value;
}
→
h1 {
    color: blue;
}
```

- **internal style sheet:** styling defined w/in `<head>`  
rules defined w/in `<style>`  
styles applied to elements in that file → useful for 1 page

```
-format: <head>
        ... metadata...
        <style>
        ...
        </style>
</head>
```

- external style sheet: rules defined in an external file  
style sheet(s) linked in `<head>`  
styler are applied to elements in all files that links that style sheet

- format: `<link rel="stylesheet" href="css file name.css">`

- ```

: style cascading order: browser default
                        external style sheets
                        internal style (inside <head> section)
                        inline style (inside an HTML element)
                        ↓ lower precedence
                        higher precedence

```

- rule precedence: most recent has precedence

→ (eg. 1 selector is defined in multiple external files): most recent file rules have precedence

→ (eg. 1 selector has multiple rules in same file): most recent rule has precedence

- override precedence rules: use !important attribute

```
- format: property: value !important;
```

## Colors

- color conventions: color names/keywords - work, but should be avoided
  - hexadecimal - common convention
  - rgb/rgba
- appropriate color use is critical to web accessibility
- don't use color alone to convey meaning → alt text, caption, etc
- check color contrast: [webaim.org/resources/contrastchecker](http://webaim.org/resources/contrastchecker)  
[wave.webaim.org](http://wave.webaim.org)

## Styling Your Text


- styling options: font
  - family
  - style
  - variant
  - sizecolor & background  
alignment  
line-height
- font-family**: different types/styles of text  
(eg. Arial, Helvetica, "Comic Sans MS")
  - format: font-family: font family name; → font-family: Arial;
  - can add alternatives, if font isn't supported: → font-family: Courier, Impact, Arial;
  - sans-serif > serif ↑ user-friendly
- custom fonts:
  - format: @font-face {
    - font-family: custom name;
    - src: url('file name.ttf');}
  - selector {
    - font-family: custom name;}
- font-style**: normal  
italic  
oblique
- font-variant**: normal  
small-caps
- font-size**: xxs, xs, s, smaller } unpopular

medium  
l, xl, xxl, larger  
pixels  
%.

- **color**: color of foreground (text)
- **background-color**: color of background
- **text-align**: left  
right  
center  
justify
- **text-transform**: lowercase  
uppercase  
capitalize  
none  
inherit
- **line-height**: adjusts the space between the lines of text  
doesn't affect font
  - format: line-height: #%;
  - (eg. line-height: 50%;): overlap
  - (eg. line-height: 200%;): spread apart
- design larger projects on paper first!

## Display & Visibility

- every element is a box → box model
- **display**: affects the layout of neighboring elements  
(ie. how to decide if the boxes should be next to, underneath each other, etc)
  - common values: inline  
block  
inline-block  
none
  - other values: table  
grid  
flexbox  
(new, not always supported)
  - complementary properties: float  
clear
- **inline**: sits next to other elements  
takes up 'just enough' width & height  
unchangeable dimensions
- **block**: forces line break / no elements next to it  
default: take up all horizontal width & 'just enough' height  
can set dimensions w/ rules
- **inline-block**: inline elements you can change the dimensions of  
block elements that can sit next to other elements
- **none**: removed from page  
still in DOM, but not visible (even to screen readers)

- **float**: reposition elements to as far left/right it can go  
elements are aware of each other & won't overlap
  - values: left  
right
- **clear**: used to keep floating elements away
  - values: left  
right  
both
- content doesn't fit in set dimensions → use overflow to determine access
- **overflow**: visible  
hidden  
scroll  
auto
- **visible**: can cause text to show up on top of other text
  - (eg: ): text goes outside the box
- **hidden**: hides anything that goes beyond bounding box
  - will cause problems if user increases font-size → content is now hidden
- **scroll**: gives horizontal & vertical scrollbars
- **auto**: adds scrollbars as needed
- **table**: table-like layout w/o using table structure
  - format: display: table;  
display: table-cell;
- **visibility**: specifies whether or not element is visible  
similar to display: none; but you can see the space it takes up
  - values: visible  
hidden  
collapse  
-only for table elements