**Cascading Style Sheets**

**resets, layouts**

# [CSS Resets](https://github.com/nycda-staff/nycda-curriculum/blob/master/lectures/wdi-css-reset/wdi-css-reset.md)

User Agent Stylesheet (UAS)

default browser styles

results in cross-browser inconsistency (each browser has its own baseline stylesheet)

CSS resets

override UAS with "standardized" styles for consistent styling

common rest stylesheets

Eric Meyer

http://cssreset.com/scripts/eric-meyer-reset-css/

normalize

https://necolas.github.io/normalize.css/

linking reset stylesheet to index.html

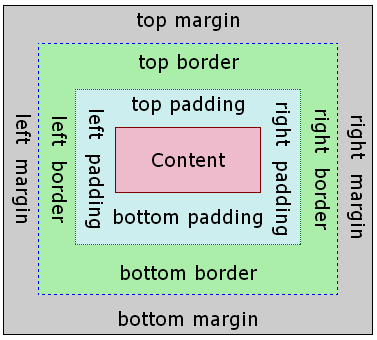
link always goes before your custom stylesheet (styles.css) link

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

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

# [CSS Layouts](https://github.com/nycda-staff/nycda-curriculum/blob/master/lectures/wdi-css-layout/wdi-css-layout.md)

## The Box Model



## Box-model syntax

clockwise edge sequence

top, right, bottom, left

used to specify margins and padding

examples:

margin: 10px 20px 5px 15px;

padding: 4px 8px 10px 6px;

shorthand

allows multiple edges to be specified as a single attribute

top/bottom/left/right

example: top/bottom/left/right = 10px

margin: 10px;

top/bottom, left/right

example: top/bottom = 10px, left/right = 20px

margin: 10px 20px;

top, left/right, bottom

example: top = 10px, left/right = 5px, bottom = 20px

margin: 10px 5px 20px;

individually specified margins and padding

each margin/padding can be specifically addressed (margin example):

margin-top: 10px;

margin-right: 20px;

margin-bottom: 30px;

margin-left: 40px;

## Box-model usage

centering content

Use auto for margin-left/margin-right to center an element

This will only work for elements with a width property.

width: 900px;

margin: 10px auto;

Use a #wrapper element to center all content on the page (all content elements go inside wrapper)

#wrapper {

width: 900px;

margin: 10px auto;

}

Border Styles

border-style -- gives style to border!

border-style: dashed; // other styles: solid (default), dotted

border-width -- sets border width

border-width: 10px;

border-color -- sets a border color

border-color: #111111;

border -- shorthand; set all 3 properties at once!

syntax:

border: [width] [style] [color]

example:

border: 1px solid red;

Box-sizing

Controls how the width of an element is calculated.

content-box -- element width = content width

border-box -- element width = content width + padding + border

box-sizing: content-box; // default

box-sizing: border-box;

## Display

the most important CSS property for controlling layout

types – block, inline, inline-block, flex, none

every element has a default display value

you can override the default display

block elements – div, h1 - h6, p, form, header, footer, section, ul/ol/dl

take up as much horizontal space as is available (width = 100% of container)

have "built-in" margins above and below (set *margin: 0* to remove)

height/width/padding/margin can be set (to override defaults)

force a new line (tolerate no adjacent HTML elements)

inline elements – em, span, input, strong

cannot have set dimensions

do not start on a new line

only take up as much width as necessary

have only left/right margins (no top/bottom)

padding is applied ineffectively

appear on the same line

inline-block elements

appear on the same line

take up as much space as required (inline)

can have set height/width/padding/margins (block)

think of inline-block as as *inline, but with dimensions*

flex

defines flexbox attributes for main container positioning (see details below)

## Height/Width

browsers *do not evaluate height/width %* unless height value is set on parent element

browsers calculate total available width as a function of browser window width

browser automatically flows contents to fill the entire width of window

NOTE: set *100% height value on body* when setting percentage heights on container elements

## Floats

Originally intended for wrapping text around other page elements ("newspaper style")

Float Syntax:

CSS

.my-float {

float: left; // other options: right, none, inherit

}

HTML

<div class="my-float">First Div Text</div>

<div class="my-float">Second Div Text</div>

clearing floats

restores "normal" behavior to following elements (e.g. blocks appear on new line)

clear value options: left, right, both

example:

.my-non-floated-element {

clear: left

}

Float effects

float: left – element floats all the way to left of container

floated adjacent siblings – will appear on same line (if container width is wide enough)

width issues

elements *will not float* if there is not enough width

block elements that follow floated elements *will not float* because...

...block elements have a built-in line-break that forces them to new line...

...*unless* they have an assigned float attribute

height issues

floated elements *do not take up vertical space* in their containing element

non-floated followers of floated element will NOT respect height

solution: clear the float:

clear: left;

float usage examples

two-column pages

image thumbnail gallery

text-flow around image ("newspaper style")

## Position

syntax:

position: relative;

position value options:

static, relative, absolute, fixed

position typical usage:

pop-ups, modals, burger-menus

position effects:

positioned elements always offset relative to *something*

offset values:

left, right, top, bottom

**position: static;**

default value; no positioning will occur

**position: relative;**

element will remain within the normal flow of the document

element will be offset with respect to its current location

example (Move the element 10px down from its current location):

<div class='move-me'>

Some content

</div>

.move-me{

position: relative;

top: 10px

}

**position: absolute;**

element will NOT remain within the normal flow of the document (may conflict/cover other elements)

elements with left, right, top, bottom attributes *must be in a positioned parent*

example (Move .child 10px down from the top of .parent):

<div class='parent'>

<div class='child'>Some content</div>

</div>

.parent {

position: relative;

height: 100px;

width: 100px;

}

.child {

position: absolute;

top: 10px

}

**position: fixed**

element will NOT remain within the normal flow of the document

element will be *offset with respect to the window*

element will *remain in place on scroll*

example (Move the element 10px from the top of the window):

<div class='move-me'>

Some content

</div>

.move-me {

position: fixed;

top: 10px;

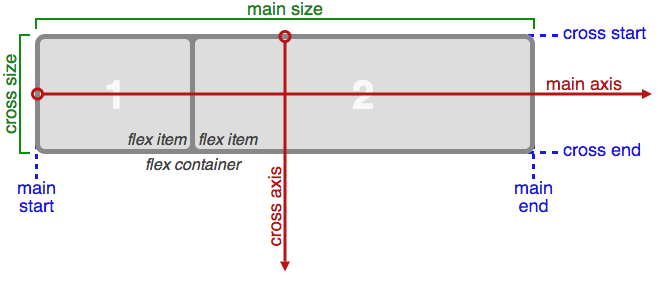
}

## Flexbox

an efficient way to position/align/distribute space among items in a container

used when item size is unknown and/or dynamic

float, clear and vertical-align have no effect on a flex item.



flex example

#flex-container {

display: flex;

flex-direction: row;

flex-wrap: nowrap;

justify-content: flex-start;

align-items: center;

align-content: space-around;

}

flex container rules (parent)

css rules (typical) additional value options for each rule

#flex-container {

display: flex;

flex-direction: row; // column | row-reverse | column

flex-wrap: nowrap; // wrap | wrap-reverse

justify-content: flex-start; // flex-end | center | space-between | space-around | space-evenly

align-items: flex-start; // flex-end | center | baseline | stretch

align-content: flex-start; // flex-end | center | space-between | space-around | stretch

}

rule explanations

flex-direction -- horizontal or vertical setting for main axis

flex-wrap -- items try to stay on one line (no-wrap); wrap allows use of next line

justify-content

flex-start -- items packed at start (left/top)

flex-end -- items packed at end (right/bottom)

centered -- items centered along direction axis

space-between -- first item at start; last item at end; middle items distributed

space-around -- single space before first and after last items; double spaces between others

space-evenly -- spaces between all items is even

align-items -- alignment for items on cross axis

baseline -- items aligned so that inner text baselines are aligned regardless of item heights

stretch -- stretch to fill container

align-content -- similar to "justify content" for cross-axis

flex items rules (children of flex container)

#flex-item {

order: <integer> // 1, 2, 3, 4, 5...

grow: <number> // 1, 2, 3

shrink: <number> // 1, 2, 3

align-self: auto // flex-start | flex-end | center | baseline | stretch

}

order -- number specifying sequence in a group of items

grow -- amount of available space item should stretch to fill

shrink -- allows items to shrink to fit on main axis

align-self -- set for individual item to over-ride general align-items property