CSS Study

Aaaahhhhh fuck.

What is the point

In this life

Original Source : W3C guidelines

Useful source: MDN – Mozilla developer Network CSS guides

Codeguide.co

For colors – neilorangepeel.com, coolers.co

Font generator – fontsquirrel.com

Core Concepts :

Selector – selects which HTML element to style

Declaration – Declares the style

<length> - can be absolute or relative

Absolute – has units of px,cm,mm

Relative – is relative to the parent element, has units of em,rem,vw.

Keywords – do not require units

CSS functions – CSS has functions that can be used as is; Eg: transform: rotate(90deg);

Width: calc(80% - 20px)

Back-img: url(myimage.png)- sets a background image

Color functions – either use keywords such as the color itself.

Or we can use rgb to specify what percentage of rgb we want to use.

rgb = ( , , ) to get colors

rgba ( , , ,) , last comma for opacity.

\*Selector - \* selects all the elements of the HTML file to set a style. Syntax is \* {….}.

ID Selector - #idname will select the specific id to act on. Id is unique and can only be used once per HTML file.

Class Selector - .classname will select the specific class to act on.

Descendant selectors : selects elements nested within other elements.

Just do p a for an element a within paragraph p. try to keep the descendant selectors down tot just three levels and not more to keep the code readable.

Grouping Selectors : instead of h1 {…} h2 {…} h3 {…} we can write h1,h2,h3 {…} if the style is the same. Keeps our code uncomplicated.

Inheritance – a child class inherits properties from its parent class.

Specificity – when there is a conflict with two properties being assigned to the same code using different selectors, there is a hierarchy of what selector takes precedence. For now this is the hierarchy : from lowest to highest.

Universal (\*) – can only override inherited styles

Type (p)

Class (.)

Id (#)

Cascade – Code is read top to bottom. If the same element is styled separately twice, the last declaration takes precedence.

!important – used to override specificity but read more before implementing.

Pseudo-class selectors – combined with other selectors using :

:link – default color blue

:visited – changes color of link once visited

:focus – applied when element comes into focus, usually when using kb

:hover – changes color if you hover over link

:active – changes color if actively press

Links must be declared in the same order as listed above because of cascading rules.

The Box Model:

The CSS box model is the basic layout of every webpage, and we need to understand it before going forward with the styling

5 properties that make up the box model: Width

height

Padding

Margin

Border

Inline – element in the line, just takes up a block of space, not the entire line, used for elements like <a>, <span>,<strong>, does not start a new line

Block – element takes up the entire line, starts a new line, used for elements like <p>, <h1>, header, <article> etc.

Display – can be used to change a element from block to inline and vice versa, can also be used to change to inline-block.

Box Model – consists of content -> Padding -> border -> margin

Five properties are height and width of content, size of padding, margin and border between the two.

Margin collapsing- The top and bottom margin are shared by elements, keep this in mind while styling

Margin can take negative values. To align the content to the middle of the page use: margin: 0 auto. This sets the content to the center by setting the top and bottom margin to 0.

Typography :

The many typefaces – Script – almost handwritten, saved for headlines and small text

Decorative – as name suggests, saved for headlines

Monospace – occupies same horizontal length, saved for code

Serif – bold characters, traditional, used often

Sans-serif – modern, used often.

Font – family – declare specific font families followed by atleast 1 generic. Eg: Helvetica, arial,san-serif.

Font-weight – adds darkness/boldness to the text. Ranges from 100-900 from lightest to darkest.

Normal = 400; bold = 700.

Lighter and bolder keyword. Inherits from parent.

Font-style – three styles: italics, oblique and normal.

Google fonts is a useful resource to get fonts for projects.

Font-size – px : absolute, must be whole numbers,

em : relative to nearest ancestor, can be decimals

rem : relative to the root, can be decimals

text–decoration – underline color style. Can be used to add or remove these three properties from text and decorate them according to our needs. Many options review MDN for more info

line-height – height of the space between two lines

Layout: Floats and Position

float – used to change the layout of the webpage from the default blocks stacked upon one

another. Can be float: left/right

clear – can be used to clear float from any one element and change it to default.

Use clear: both;

overflow – used to take care when the content flows out of the block intended. Can be

used in 3 ways.

1. Hidden – cuts of the overflow part, but we cannot access it in any way, useful for images, not so much for text.

2.auto – adds a vertical scroll to the overflowing content

3.scroll – adds a vertical and horizontal scroll

The box model fix – add this snippet of code to make life easier : Add to all our projects.

Html {

box-sizing: border-box;

}

\*, \*:before, \*:after {

box-sizing: inherit;

}

Position – another way to change position of blocks. Study this bloody borin.

Z index – a z index value changes the position of blocks which are stacked on top of each other. Can give priority to different stacks using z index value.

Layout: Flexbox and Grid

Flexbox: good for layout in one dimension

Grid: good for layout in 2 dimensions

Flex-container: the parent element

Flex-items: the child elements

: items are placed along the main axis, default is horizontal

: axis can be changed to vertical using flex-direction? I think

: axis has start and end point called main start and main end for main axis and cross start and cross end for cross axis

To use flexbox we must first define flex container: <div class=”container”>

:<div> flex item </div>

:<div> flex item </div>

:<div> flex item </div>

And then we use display to display the flexbox: .container {

Display: flex; OR display: inline-flex;

Flex-direction: determines the dir of main axis: row:ltr

: row reverse:rtl

: column: changes main axis to vertical dir

: column reverse

Items usually smaller than size of container, then there is space left.

If the size of items is larger than container then items shrink to fit into the container.

Flex-wrap: wraps the items to the container, default is nowrap

:wrap-instead of fitting in 1 line, items will wrap to the container in a second line.

:wrap-reverse – changed the order of wrapping

Shorthand to apply wrap: flex-flow: column wrap for flex-direction column and flex-wrap wrap

Sizing - flex-sizing: sets intial size of flex items

flex-grow: how items will expand if extra space in container

flex-shrink: how items will shrink if not enough space in container

shorthand- flex: grow shrink basis

must be applied to each flex item instead of container : .flex-item { flex: 1 1 100px}

Alignment – justify-content: aligns items on the main axis

Align-items: aligns items on cross axis

Syntax: .box {

Display: flex;

Justify-content: center;

Align-items: center; }