**CSS**

CSS RULE = it needs a selector and a declaration. A declaration contains a property and a value.

**COLOR**

Color: can be expressed in rgb (values for red, green, and blue expressed as numbers), color name or hex value (hexadecimal code).

Hue: color

Saturation: how much gray is in the color. Max. sat.= no gray; Min. sat=gray

Brightness: how much black is in the color. Max. bright.= no black; bright. sat=black

Lightness: how much white (lightness) or black (darkness) is in the color. Max. light.= white; Min. light=black

hsla colors: hue, sat, light, alpha.

/\*…\*/ \*to make comments in css

color:

background-color:

opacity: (0-1) \*to add opacity to an element and its child elements.

rgba: \*to add a fourth value in rgb nomenclature that indicates opacity.

hsl or hsla(hue(angle), sat.%, light.%, transparency(0-1))

**TEXT**

* serif = with details
* sans-serif= no details
* monospace = letters have same width
* font-family: \*type of font
* font-size: (px,%,ems)
* font-weight: \*bold or normal
* font-style: \*italic, oblique
* text-transform \*uppercase. Lowercase or capitalize
* text-decoration: \*none, underline, overline, line-through, blink
* line-height: \*vertical gap between lines (leading) .(ems)
* letter-spacing: \*space between letters (kerning) .(ems)
* word-spacing: \*gap between words.(ems)
* text-align: \*left, right, center, justify
* vertical-align: \*align text vertically.
* text-indent: \*indent the first line of text of an element
* text-shadow:
* :first-letter \*pseudo-element
* :first-line \*pseudo-element
* :link \*set style to unvisited links. (pseudo-class)
* :visited \*set style to visited links. (pseudo-class)
* :hover \*set style to hovered links. (pseudo-class)
* :active \*set style to active links, suchs as clicks. (pseudo-class)
* :focus \*to elements that you can interact with.. (pseudo-class)

**BOX DIMENSIONS**

* width:
* height:
* min-width: \*to adjust for different screen devices
* max-width: \*to adjust for different screen devices
* min-height: \*to adjust for different screen devices
* max-height: \*to adjust for different screen devices
* overflow: (hidden or scroll) \*to adjust contained text that occupies a bigger space than its container.
* border-width:
* border-style:
* border-color:
* border: \*put border width, style and color in one property.
* padding: \*space between the content of an element and its border.
* margin: \*controls the gap between boxes. To center a box set the left and right margins to auto.
* display: (inline, block, inline-block, none) \*allows you to turn an inline element into a block level element and vice versa.
* visibility: (hidden, visible) \* allows you to hide boxes from users but leaves the space where the element would have been
* border-image:
* box-shadow:
* border-radius:

**LISTS, TABLES AND FORMS**

* list-style-type \*to control shape and style of bullet points
* list-style-image: url(“”) \*to specify an image to act as a bullet point
* list-style-position: (outside, inside) \*to specify the position of bullet points in the text.
* list-style: \*to write the style, image, and position in any order.
* empty-cells: (show, hide, inherit) \*to show borders of empty cells
* border-spacing: \*to control distance between adjacent cells
* border-collapse: \*to put together adjacent cells
* cursor: \*to control the appearance of the mouse over some element

**LAYOUT**

* position: static \*normal flow. It affects the position of surrounding elements.
* position: relative \*the element is relative to the position of where it would have appeared if it had been in normal flow. It does not affect the position of surrounding elements.
* position: absolute \*the element position is relative to the containing element. It does not affect the position of surrounding elements.
* position: fixed \*it is fixed to the browser’s window. You will always see these elements even if you scroll down. It does not affect the position of surrounding elements. (left, right, top, bottom offset values)
* z-index: (#) \*when boxes overlap, you can determine which box should appear over the other one. Just as bring to the front in word.
* float: \*to move an element either to the far right or left of its containing element.
* clear: \*allows you to say that no element, within the same containing element, should touch the left or right hand sides of a box. It will appear on the bottom of the last floated element.

**IMAGES**

* background-image: url(“”)
* background-repeat: \*(repeat, repeat-x, repeat-y, no-repeat)
* background-attachment: \*(fixed, scroll), with fixed it stays in the same position on the page
* background-position: \*when an image does not repeat you can set the position of it on its container using two values, one horizontal and the other vertical. (used for CSS Sprites)
* background: \*shorthand for the following properties in order = background-color, background-image, background-repeat, background-attachment, background-position.

**RESPONSIVE WEB DESIGN**

* @media screen and (max-width: 900px) \*responsive css for 900px window or smaller
* @media screen and (min-width: 900px) \*responsive css for 900px window or bigger
* <meta name="viewport" content="width=device-width, initial-scale=1.0"> \*for responsive web design.
* <link rel="stylesheet" href="responsive.css">
* box-sizing: border-box; \*elements should have padding and border included in the element's total width and height specifications.

This markup is for when dealing with floats whose containing element is not big enough for them. With the pseudo-class :after, a dot is included with no visibility and with the clear property (so that it appears below the last floated element) so that the containing element can put on enough space for floated elements without having to add-up any more html markup (create an html element with no content in it). The second declaration is just for browser adaptability of the first declaration.

|  |  |
| --- | --- |
| 1 | .clearfix:after { |
| 2 | **content**: "."; |
| 3 | **display**: block; |
| 4 | **height**: 0; |
| 5 | **clear**: both; |
| 6 | **visibility**: hidden; |
| 7 | } |
| 8 |  |
| 9 | .clearfix {**display**: inline-block;} |
| 10 |  |
| 11 | /\* Hides from IE-mac \\*/ |
| 12 | \* html .clearfix {**height**: 1%;} |
| 13 | .clearfix {**display**: block;} |
| 14 | /\* End hide from IE-mac \*/ |

**TIPS**

how to center with CSS?

1.

Text-align: center; (targeted at the container of block elements)

display: inline-block;(for block elements so that they can be treated as text)

2.

Margin: 0 auto; (distributes evenly the margin on the left and right of the element)

3. \*you have to know the exact width and height dimensions of the block element to pull this off{

Position: absolute;

left: 50%;

margin-left: -200px; (half the width of your block element or img)

top:50%;

margin-top: -200px;}

.container {position: relative; (of the containing element)}

4. \*you do not have to know the exact width and height dimensions of the block element to pull this off {

Position: absolute;

left: 50%;

top:50%;

+transform(translate(-50%,-50%));}

.container {position: relative; (of the containing element)}

**This markup if for navigation bars, in order for its list elements to take the full width of the layout.**

/\* this selector could be div#nav instead of nav depending on which tag you wrapped the ul in \*/

nav {

display: table;

width: 100%;

  border-collapse: collapse;

  border: none;

}

nav ul {

  display: table-row;

 }

nav li {

  display: table-cell;

  margin: 0;

}