**HTML/CSS NEW ELEMENT DEFINITIONS AND APPLICATION**

**HTML**

**CSS**

**-LINE HEIGHT:**

SYNTAX: Line-height: 10px;

DEFINITION: Creates more space between your lines, so it makes your text more readable.

APPLICATION: You can specify this in px, em, or %

EXAMPLE: 10px, 1.5em, 10%

**-CUSTOM CURSOR:**

SYNTAX: cursor: pointer;

CURSOR PROPERTY VALUE LIST:

-default

-inherit

-auto

-URL

-crosshair

-pointer

-move

-text

-help

-hand

-no-drop

-alias

-progress

-wait

-grab

-grabbing

-copy

-cell

-spinning

**-HOVER:**

SYNTAX: class-name:hover{

Background: grey;

Padding: 20px;

}

DEFINITION: Basically creates a condition. It gives your chosen class or id all the values you choose for it, whenever you hover over that particular element

APPLICATION: For example, you can use this to make a button change colors, or bigger when people hover over it.

**-BACKGROUND-SIZE:** The background-size property in CSS is one of the most useful — and most complex — of the background properties. There are many variations and different syntaxes you can use for this property, all of which have different use cases.

USE THIS SITE TO LEARN ALL ABOUT THIS. THIS IS REALLY IMPORTANT:

https://css-tricks.com/almanac/properties/b/background-size/

BASIC-SYNTAX: background: url(greetingimage.jpg);

Background-size: 300px 100px;

KEYWORDS: Auto, cover, and contain

-AUTO: tells the browser to automatically calculate the size based on the actual size of the image and the aspect ratio.

-COVER:  tells the browser to make sure the image *always* covers the entire container, even if it has to stretch the image or cut a little bit off one of the edges.(How much of the page is covered by the image depends on the height percentage you give it.)

-CONTAIN: contain, on the other hand, says to *always* show the whole image, even if that leaves a little space to the sides or bottom.

**POSITIONS:** There’s 4 types of positions you can have: static, relative, absolute, and fixed. You have to understand exactly how each of them behave, in order to be able to command them at will.

-STATIC: Does nothing. It’s the default position, It doesn’t move at all.

-RELATIVE: The image floats up to the top of the z index, but it doesn’t affect the flow of the DOM. Or in other words. All other elements around it respect the spot of the element you floated as if it had never moved. Picture a Buddhist levitating into thin air, but everyone around it reserves his spot for when he comes down.

-ABSOLUTE: The effects of the absolute positioning are ripped outside of the dom, as if the object had never existed. Picture that same Buddhist monk levitating, except this time he’s in his spirit form, so no one can see him. Therefore, all the elements, or other monks behave as if he wasn’t around.

-ANCESTOR WITH DECLARED POSITION: When you declare a directional property on an absolutely positioned element, it will be positioned relative to it’s nearest parent with a declared position statement. If you don’t have one, it will be declared relative to the dom.

-DECLARING OPPOSING POSITIONS: If you declare opposing positions such as left followed by right…

EXAMPLE:

.son{

position:absolute;

left:10px;

right: 10px;

}

…The element is going to expand to meet both demands.

-FIXED: Similar to the absolute position, the fixed element causes everything to collapse around it, meaning, that it has no width of height, but it’s even more so than the absolute, because the element will not move even when you scroll. A fixed position will never respect any of it’s parents. It’s always positioned relative to the document window: Not even the dom, not even the body, or the HTML element. Similar to the absolute position, you can expand the element by calling contradicting directionals(This is how people make headers that stay fixed on the page, and everything scrolls underneath it)

**DISPLAY PROPERTY:** We’ll discuss the difference between inline and block elements, followed by their default state.

-INLINE: They sit inline with each other, just like the words on a text, sit next to each other

-RULES: vertical margins, widths, heights, and other values are not affected by CSS manipulation in inline elements. If you add vertical padding, it will show you the added padding, but nothing around it will move.

IMAGES BY DEFAULT ARE INLINE

-BLOCK: Block elements are affected by pretty much all CSS manipulations we add to them, such as: padding, margins, width, height, etc.

-h1 through h6, divs, paragraph tags are block by default

-INLINE-BLOCK: Best of both worlds. You can give it all the properties of a block element, while it’s still inline.

**Z-INDEX:** Z-index is super easy. Basically you add the z-index property, and declare a number. The higher the number, the higher above on the z-index the element is going to be. Apparently z-index only works on relative, absolute, or fixed positions.

SYNTAX: z-index: 3;

**FLEXBOX:** ?