16/01/2024 - Building a set of colored markers.

① `<!DOCTYPE html>`
`<html lang="en"></html>`

② Nesting head element & body element within html element.
`<head></head>`
`<body></body>`

③ Adding a title element → Colored Markers.

```
<head>
  <title> Colored Markers
  </title>
</head>
```

(?) title element gives search engines extra info about the page. It also displays the content of that title element in 2 more ways: 1) in the title bar when the page is open; 2) in the browser tab for the page when you hover on it. Even if that tab is not active, once you hover on the tab, the title text is displayed.

④
```
<head>
  <meta charset="utf-8">
  <title> Colored Markers </title>
</head>
```

utf-8 is a universal character set that includes almost every character from all human languages.

⑤ Adding another self-closing meta element. Give it a `name` attribute set to `viewport` and a `content` attribute set to `width=device-width, initial-scale=1.0` so your page looks the same on all devices.

`<meta name="viewport" content="width=device-width, initial-scale=1.0">`

⑥ Adding h1 element with the text CSS Color Markers.

⑦ Linking index.html to styles.css file.

```
<head>
  <meta charset>
  <meta name....>
  <title></title>
  <link rel="stylesheet" href="styles.css">
</head>
```

⑧ (styles.css) adding ↴

```
h1 {
  text-align: center;
}
```

⑨ (index.html) adding a div element and set its class attribute to container. Make sure the div element is below the h1 element.

`<div class="container"></div>`

⑩ (index.html) adding another div element within the div element previously & give it a class of marker. →

```
<div class="container">
  <div class="marker"></div>
</div>
```

⑪ (styles.css)

```
.marker {
  background-color: red;
}
```

⑫ (styles.css)

```
.marker {
  background-color: red;
  height: 25px;
  width: 200px;
}
```

(13) (styles.css) sets its margin property to auto.
This setting sets margin-top/right/bottom/left to auto. It's called as the margin shorthand property ;)

(14) (index.html) adding another (two more, actually) elements and give them each a class of marker. [inside the container div].

```
<div class = "container">
  <div class = "marker"></div>
  <div class = "marker"></div>
  <div class = "marker"></div>
</div>
```

(15) (styles.css) setting the margin property to 10px auto.

```
margin : 10px auto;
```
(margin: shorthand property; 10px: value1; auto: value2)

→ when the shorthand property has 2 values, it sets margin-top & margin-bottom to the 1st value, and margin-left & margin-right to the 2nd value.

(16) (index.html) adding the class one to the 1st marker div element.

```
<div class = "one marker"></div>
```

(17) (styles.css) removing the background-color property and its value from the .marker CSS rule.

(18) (styles.css) creating a new CSS rule that targets the class one and set its background-color property to red.

```
.one {
  background-color: red;
}
```

(19) (index.html) adding the class two to the 2nd marker div, and add the class three to the third marker div. (see step 16).

(20) (styles.css) adding background-color properties for the 2nd & 3rd markers: green and blue.

(21) (styles.css) Main color models: the additive RGB and the subtractive CMYK (used in print) (used in electronic devices)

create a new CSS rule that targets the class container and set its background-color to black with rgb(0, 0, 0).

```
.container {
  background-color: rgb(0,0,0);
}
```

(22) (styles.css) adding a function. A function is a piece of code that can take an input and perform a specific action.

in the .one CSS rule, replace the color keyword red with the rgb function.

rgb(red, green, blue); → the format.

```
.one {
background-color : rgb(255, 0, 0);
}
```
→ this will be red.

(23) (styles.css) # use the rgb function to set the background-color to the max value for green in .two CSS rule, and max value for blue in .three CSS rule.

```
.two {
background-color : rgb(0, 255, 0);
}

.three {
background-color : rgb(0, 0, 255);
}
```

(24) (styles.css) adjusting the color of the green marker so it will have a darker shade. →

```
.two {
background-color : rgb(0, 127, 0);
}
```

(25) (styles.css) in the .container CSS rule, use the shorthand padding property to add 10px of top & bottom padding and 0 px of left & right padding.

padding-top : 10 px ; ... etc.

(26) (styles.css) in the additive RGB color model, primary colors are colors that, when combined, create pure white. But for this to happen, each color needs to be at its highest intensity.

set your green marker to the max value of 255. (see step 24).

(27) (styles.css) [now you have primary RGB colors, yeay!] let's combine them. # set the red, green, and blue values to the max for the rgb function in the .container rule. (see step 21 for the reference).

(28) (styles.css) secondary colors are the colors you get when you combine Primary colors. # update the rgb function in the .one CSS rule to combine pure red and pure green into yellow → set the rgb into rgb(255, 255, 0) [see step 22 as reference].

(29) (styles.css) .two → background-color : rgb(0, 255, 255) [see step 23 as reference] → changing into cyan.

(30) (styles.css) .three → rgb(255, 0, 255) [see step 23 as reference] → changing into magenta. WOW !!!

(31) (styles.css) # Learn how to create tertiary colors ☺ [primary + secondary]

.one → rgb(255, 127, 0) → ORANGE ☺ → why? bcs you ↑ red & ↓ green. and its because orange = red + yellow.

(32) (styles.css)

Create tertiary color spring green → .two → rgb(0, 255, 127)

(33) (styles.css) # create violet → .three → rgb(127, 0, 255)

(34) There are 3 more tertiary colors :
chartreuse green (green + yellow), azure (blue + cyan), and rose (red + magenta).

- one → rgb (127, 255, 0) ⇒ chartreuse green
- two → rgb (0, 127, 255) ⇒ azure
- three → rgb (255, 0, 127) ⇒ rose.

(35) (styles.css) → change all into black again.
rgb (0, 0, 0)

(36) (styles.css)
[understanding a color wheel]
a color wheel is a circle where similiar colors or hues are near each other, and different ones are further apart. For example, pure red is between the hues rose & orange.
Two colors that are opposite from each other on the color wheel are called complementary colors. If 2 complementary colors are combined, they produce gray. But when they are placed side-by-side, these will produce strong visual contrast and appear brighter.

- one → rgb (255, 0, 0) → pure red
- two → rgb (0, 255, 255) → cyan.

(37) (styles.css)
[understanding a color wheel pt.2]
Notice that red & cyan are very bright right next to each other. This contrast can be distracting & hard to read.
It's better to choose one color as the dominant color & use its complementary color as an accent to bring attention to certain content on the page.

```
h1 {
  background-color : rgb (0, 255, 255)
}
```

(38) (styles.css)

- one → rgb (0, 0, 0)
- two → rgb (255, 0, 0)
- three → rgb (0, 0, 0)

it will look like this : black / red / black

your eyes are naturally drawn to the red color in the center.
When designing a site, you can use this effect to draw attention to important headings, buttons, or links.

(39) (styles.css) # set to black again for all.

(40) (styles.css) removing background-color property.

(41) (index.html) adding other details to the markers.
change the class one to red.

(42) (styles.css) update the .one CSS rule to target the new red class.

one .////// { → red .////// {
background-color : rgb (0,0,0) → background-color : rgb (0,0,0)
}

(43) .red → rgb (255, 0, 0) (styles.css)

(44) (index.html) changing the class two to green in the div elements.
[see step 41]

(45) (styles.css) changing the class selector .two → green & .three → blue.
[see step 42]

(46) (styles.css) using hexadecimal (hex) values.
Hex values start with a # character & take six chars from 0-9 and A-F. The first pair of chars represent red, the 2nd represent green, and the 3rd represent blue. e.g. #485320.
change the .green class selector → values : #00ff00 (green).

(47) (styles.css)
[HEXADECIMAL VALUES] 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
With hex colors, 00 is 0% of that color, FF means 100%.
So, #00FF00 translates to 0% red, 100% green, and 0% blue. It is same as rgb (0, 255, 0).
lower the green's intensity by setting the .green → #007F00

(48) (styles.css)
[understanding HSL color model] * HSL = hue, saturation, & lightness.
The CSS HSL function accepts 3 values : a number from 0 to 360 for hue, a percentage from 0 to 100 for saturation and a percentage from 0 to 100 for lightness. If you imagine a color wheel, the hue red is at 0 degrees, green at 120 degrees, and blue is at 240 degrees.
Saturation → 0% (gray), 100% (pure).
Lightness → 0% (black), 50% (neutral), 100% (white).
set the .blue
```
.blue {
  background-color : hsl (240, 100%, 50%)
}
```

(49) (styles.css)
[using a color transition / gradient, on an element]
The CSS linear-gradient function lets you control the direction of the transition along a line.
! linear-gradient function actually creates an image element, and is usually Paired with the background property.
change the background-color property to background.

```
.red {
  background: rgb(255, 0, 0);
}
```

(50) (styles.css)
The linear-gradient function is very flexible. Here's the basic syntax:
linear-gradient (gradientDirection, color 1, color 2....);
gradientDirection → direction of the line used for transition.
color1, color2 → color arguments, which are the colors that will be used.
(no space) These can be any type of color, including color keywords, hex, rgb, or hsl.

```
.red {
  background: linear-gradient(90deg);
}
```

(51) (styles.css) adding rgb function in the linear-gradient function.

```
.red {
  background: linear-gradient(90deg, rgb(255, 0, 0));
}
```

(52) (styles.css) you won't see gradient because the linear-gradient function needs at least 2 color arguments to work.
background: linear-gradient(90deg, rgb(255, 0, 0), rgb(0, 255, 0));
[see step 51].

(53) (styles.css) adding color3 (argument) with pure blue. [see step 52].

(54) (styles.css) color-stops → allow you to fine-tune where colors are placed along the gradient line. They're length unit like px or percentages that follow a color in the linear-gradient function.
ex: linear-gradient(90deg, red 90%, black);
# add a 75% color stop after the 1st red color argument.
background: linear-gradient(90deg, rgb(255, 0, 0) 75%, rgb(0, 255, 0), rgb(0, 0, 255));

(55) (styles.css) changing the degree into 180° [see step 50].

(56) (styles.css) setting the color stops: red 0%, green 50%, blue 100%.
red → rgb(255,0,0) #fff 0%, rgb(0,255,0) 50%, rgb(0,0,255) 100%.
↓ green ↓ blue

(57) (styles.css) changing the first color argument into: rgb(122, 74, 14)

(58) (styles.css) modifying the second color argument (the pure green one).
update the rgb function so the value of red: 246, green: 62, blue: 113.
[see step 56].

(59) (styles.css) modifying the third color argument (the pure blue one) with the value (167, 27, 27). [see step 56].

(60) (styles.css) changing the background-color property to background.
[in the .green css rule].

(61) (styles.css) using hex color codes.

```
.green {
  background: linear-gradient(180deg, #556B0D);
}
```

(62) (styles.css) adding another value next to #556B0D
background: linear-gradient(180deg, #556B0D, #71F53E);

(63) (styles.css) adding third value: #11CC31.

(64) (styles.css) [even without the color stops, you might have noticed that the colors for the green marker transition at the same points as the red marker. The first color is at the start 0%, middle 50%, last 100%.
remove the percentages of .red CSS rule.

(65) (styles.css) [if no gradientDirection argument is provided to the linear-gradient function, it arranges colors from top to bottom, or along a 180° line, by default.
remove the gradientDirection (the ones that have degrees).

(66) (styles.css) [for the .blue CSS rule] → removing & changing the background-color property [see step 60].

(67) (styles.css) for the .blue, use linear-gradient funotion & pass the hsl function with the values 186, 76%, 16% (hsl).

```
.blue {
  background: linear-gradient(hsl(186, 76%, 16%));
}
```

(68) (styles.css) adding the 2nd color argument for the .blue CSS rule.

```
background: linear-gradient(hsl(186, 76%, 16%), hsl(223, 90%, 60%));
```

(69) (styles.css) adding the 3rd color argument `hsl(240, 56%, 42%);`

(70) (index.html) building the marker sleeves. Start with the red marker

inside the red marker div element, create a new div element & give it a class of sleeve.

```
<div class = "marker red">
  <div class = "sleeve"></div>
</div>
```

(71) (styles.css) creating a new CSS rule that targets the class sleeve. Set the width property: 110 px and height: 25 px.

```
.sleeve {
  width: 110px;
  height: 25px;
}
```

(72) (styles.css) → adding background-color: White in .sleeve.

(73) setting the opacity property to 0,5 [0.5, actually,]

opacity → describes how opaque (transparency).

0 or 0% → completely transparent.

1.0 or 100% → completely opaque.

(74) (styles.css) [another way to set the opacity for an element is with the alpha channel]. # remove the opacity property & its value.

(75) (styles.css) [adding an alpha channel to an rgb color using rgba function].

Rgba function works just like the rgb function, but takes one more number from 0 to 1.0 for the alpha channel:

```
rgba(redValue, greenValue, blueValue, alphaValue);
```

You can also use an alpha channel with hsl and hex colors.

in the .sleeve rule, use the rgba function to set the background-color property to pure white with 50% opacity.

```
.sleeve {
  width: 110px;
  height: 25 px;
  background-color: rgba(255, 255, 255, 0.5);
}
```

(76) (index.html) adding a new div with the class "cap" before the sleeve div element. [see step 70 for reference].

```
<div class = "cap"></div>
```

(77) (styles.css) creating new CSS rule.

```
.cap {
  width: 60px;
  height: 25 px; }
```

(78) (styles.css) [the sleeve seems disappeared]. Why? Because they are 2 block elements. To position 2 div elements on the same line, set their display properties to inline-block. Create a new rule target both the cap & sleeve classes:

```
.cap, .sleeve {
  display: inline-block;
}
```

(79) (styles.css) adding the border-left-width: 10px; in the .sleeve CSS rule. [see step 75 for reference].

(80) [styles.css] adding the border-left-style: solid; after that.

(81) [styles.css] adding the border-left-color: black; after step 80.

(82) (styles.css) the border-left shorthand property. Syntax:
border-left: width style color; [without any commas].
border-left: 10px solid black; [see step 79-81].

(83) (styles.css) changing the value into:
border-left: 10px double black;

(84) (styles.css) changing the opacity.
! We're still talking about the .sleeve CSS rule, okay?
So, step 75 + 79 + 80 + 81 + 82 + 83 + 84 becomes:

```
.sleeve {
  width: 110px;
  height: 25px;
  background-color: rgba(255, 255, 255, 0.5);
  border-left: 10px double rgba(0, 0, 0, 0.75);
}
```

(85) (index.html) repeat step 70 by adding div class "sleeve" and repeat step 76 by adding div class "cap".
Urutannya → marker red/green/blue → div class "cap" → div class "sleeve".

(86) (styles.css) adding a slight shadow by applying box-shadow property.
Here's the basic syntax:
box-shadow: offsetX offsetY color;
Here's how the offsetX and offsetY values work:
- both offsetX and offsetY accept number values in px and other CSS units.
- a positive (+) offsetX value moves the shadow right and a negative (-) value moves it left.
- a (+) offsetY value moves the shadow down and a (-) value moves it up.
- if you want a value of zero (0) for any or both offsetX and offsetY, you don't need to add a unit.

The height and width of the shadow is determined by the height and width of the element it's applied to. You can also use an optional spreadRadius value to spread out the reach of the shadow.

add `box-shadow: 5px 5px red;` in .red CSS rule.

(87) change the box-shadow property and change into:
`box-shadow: -5px -5px red;` in .red CSS rule.

(88) (styles.css) adding blurRadius value for the box-shadow property.
`box-shadow: offsetX offsetY blurRadius color;` → new syntax
.green CSS rule.
`box-shadow: 5px 5px 5px green;`

(89) (styles.css) adding spreadRadius value for .blue CSS rule.
Syntax: `box-shadow: offsetX offsetY blurRadius spreadRadius color;`
`box-shadow: 0 0 0 5px blue;`

(90) (styles.css) [go back to .red CSS rule]
change the value(s) of box-shadow property.
`box-shadow: 0 0 20px 0 red;`

(91) (styles.css) adding the rgba function in the box-shadow of .red CSS.
`box-shadow: 0 0 20px 0 rgba(83, 14, 14, 0.8);`

(92) (styles.css) updating the values for the box-shadow property.
.green CSS → `box-shadow: 0 0 20px 0 green;`
.blue CSS → `box-shadow: 0 0 20px 0 blue;`

(93) (styles.css) changing the color with a hex color code of box-shadow property in .green CSS rule.
`box-shadow: 0 0 20px 0 #3B7E20CC;`
↳ alpha channel.