20/02/2024

[City Skyline]

1. Adding the DOCTYPE.
2. HTML lang = "en".
3. Adding head & body tags within the html element.
4. [within the head] meta charset, title, and link element.
5. (styles.css) You can target everything with an asterisk.

```
* {
  border: 1px solid black;
}
```

6. (styles.css) [cont'd]

```
box-sizing: border-box;
```

7. (styles.css)

```
body {
  height: 100vh;
  margin: 0;
  overflow: hidden;
}
```

→ to hide any scroll bars that appear when something extends past the viewport.

8. (index.html) [within the body]

```
<div class="background-buildings"></div>
```

9. (styles.css)

```
.background-buildings {
  width: 100%;
  height;
}
```

10. (index.html) [within step 8]

```
<div class="bb1"></div>
```

(styles.css)

```
.bb1 {
  width: 10%;
  height: 70%;
}
```

11. (index.html) [within the bb1, step 10] adding 4 (↓ div) elements with classes = bb1a, bb1b, bb1c, bb1d.

12. (styles.css) [adding width & height for bb1a - bb1d].

13. (styles.css) [I don't get it!] → I don't understand the instruction 😔

Instruction: center the parts of your building by turning the (.bb1) element into a flexbox parent. Use the (flex-direction) and (align-item) properties to center the children.

```
.bb1 {
width: 10%;
height: 70%;
display: flex;
flex-direction: column;
align-items: center; }
```

(14.) (styles.css) [it's getting hard].

Ins: now you have something that is resembling a building. You are ready to create your 1st variable. Variable declarations begin with 2 dashes (--) and are given a name and a value like this:

```
--variable-name: value;
```

In the rule for the bb1 class, create a variable named --building-color1 and give it a value of #999.

[within the .bb1] `--building-color1: #999;`

(15) (styles.css) [what's this ?!.]

To use a variable, put the variable name in parentheses with var in front of them like this: `var(--variable-name)`. Whatever value you gave the variable will be applied to whatever property you use it on.

Add the variable --building-color1 you created in the previous step as the value of the background-color property of the .bb1a class.

[within the .bb1a]

```
background-color: var(--building-color1);
```

(16) (styles.css). applying this ↑ for .bb1b, .bb1c, and .bb1d.

(17) (styles.css) changing the value of variable from #999 (see step 14) to #aa80ff.

(18) (index.html) # nest 3 new div elements.

```
<div class="bb2"></div>
<div class="bb3"></div>
<div class="bb4"></div>
```

(19) (styles.css) # give the .bb2 - .bb4 width & height (in percentage).

(20) (styles.css) [I DON'T GET IT !!!.].

The buildings are currently stacked on top of each other. Align the buildings by turning the .background-buildings element into a flexbox parent. Use the align-items and justify-content properties to evenly space the buildings accross the bottom of the element.

```
display: flex;
align-items: flex-end;
justify-content: space-evenly;
```

(21) (index.html) Adding 5 new empty div elements :

2 before .bb1

1 between .bb3 & .bb4

2 after .bb4

(22) (styles.css) creating a new variable below your --building-color1 variable, give it a value of #66cc99, and set it as the background-color of .bb2.

--building-color2: #66cc99; → background-color : var(--building-color2);

(23) (styles.css) [what is this omg]

That didn't work. You should add a fallback value to a variable by putting it as the 2nd value of where you use the variable like this : var(--var-name, fallback-value).

The property will use the fallback value when there's a problem with the variable. Add a fallback value of green to the background-color of .bb2.

background-color : var(--building-color2, green);

(24) (styles.css)

[within .bb1] --building-color3 : #ccG699;

[within .bb3] background-color : var(--building-color3, pink);

(25) (styles.css) [what the.....]

That didn't work, because the variables you declared in .bb1 do not cascade to the .bb2 and .bb3 sibling elements.

That's just how CSS works. Because of this, variables are often declared in the :root selector.

This is the highest level selector in CSS; putting your variables there will make them usable everywhere. Add the :root selector to the top of your stylesheet, and move all your variable declarations there.

```
:root {
  --building-color1 : #aa88ff;
  --building-color2 : #66cc99;
  --building-color3 : #ccG699;
}
```

(26) (styles.css) removing the fallback values.

(27) (styles.css) [within the :root]

```
--building-color4: #538cc6;
```

[within the .bb4]

```
background-color: var(--building-color4);
```

Najib ☺

(28) (index.html) creating a new div element.

```
<div class="foreground-buildings"></div>
```

(29) (styles.css) [cont'd].

```
.foreground-buildings {
  width: 100%;
  height: 100%;
  Position: absolute;
  top: 0;
}
```

(30) (index.html, cont'd from step 28). Nesting 6 div elements with classes of fb1 (foreground building 1) - fb6.

(31) (styles.css) adding width & height values for .fb1 - .fb6.

(32) (styles.css) setting the .foreground-buildings.

(33) (styles.css) optimizing the code

```
.background-buildings, .foreground-buildings {
  width: 100%;
  height: 100%;
  display: flex;
  align-items: flex-end;
  justify-content: space-evenly;
  position: absolute;
  top: 0;
}
```

(34) (styles.css) removing the .foreground-buildings entirely.

(35) (styles.css) just customizing.

(36) (index.html) adding some empty div elements.

→ Waterdrop

(37) (styles.css) [within the .fb4 and .fb5] adding these:

```
position: relative;
left: 10%;
```
↓ .fb4

```
position: relative;
right: 10%;
```
↓ .fb5

(38) (styles.css) Adding comments.

```
/* BACKGROUND BUILDINGS - "bb" stands for "background building"*/
```
↓ space [pay attention to the format].

(39) (styles.css) [within the :root { }]

```
--window-color1: black;
```

(40) (styles.css) Adding gradient colors. They are applied to the background property and the syntax looks like this:

```
gradient-type(
  color1,
  color2
);
background: linear-gradient (var(--building-color1), var(--window-
  color1));
```

(41) (styles.css) creating a new class selector. → .bb1-window.

(42) (index.html) adding the new bb1-window class to the .bb1a .bb1b .bb1c elements.

```
<div class="bb1">
  <div class="bb1a bb1-window"></div>
  <div class="bb1b bb1-window"></div>
  <same like above>
</div>
```

(43) (styles.css) removing the height & background-color property

(44) (styles.css) adding gradient colors → for .bb1d

```
background: linear-gradient(
  orange,
  var(--building-color1),
  var(--window-color1)
);
```

(45) (styles.css) just removing the background-color property.

(46) (styles.css) specifying the intensity of the gradient.

```
gradient-type (
color 1,
color 2  20%,
color 3
);
```

```
background: linear-gradient(
orange,
var(--building-color1)  80%,
var(--window-color1)
);
```

(47) (styles.css) removing the orange color & change the 80% to 50%.

(48) (index.html) adding 2 div elements within .bb2

```
<div class="bb2a"></div>
<div class="bb2b"></div>
```

(49) (styles.css) adding width & height properties for .bb2b.

(50) (styles.css) adding new variable in :root. [after --window-color1].

```
--window-color2:  #8cd9b3;
```

(51) (styles.css) gradient transitions often gradually change from one color to another. You can make the change a solid line like this:

```
linear-gradient(
  var(--first-color)  0%,
  var(--first-color)  40%,
  var(--second-color)  40%,
  var(--second-color)  80%
);
```

add a linear-gradient to .bb2b that uses --building-color2 from (0%) to (6%) and --window-color2 from (6%) to (9%)

```
background: linear-gradient(
  var(--building-color2)  0i,
  var(--building-color2)  6%,
  var(--window-color2)  6%,
  var(--window-color2)  9%
);
```

(52) (styles.css) [cont'd]
changing the linear-gradient into repeating-linear-gradient.

(53) (styles.css) using some tricks to make the .bb2a section into a triangle at the top of the building.

(54) (styles.css) # create & add the following properties to .bb2a

```
margin:  auto;
width:  5vw;
height:  5vw;
border-top:  1vw solid  #000;
border-bottom:  1vw solid #000;
border-left:  1vw solid #999;
border-right:  1vw solid #999;
```

(55) (styles.css) [cont'd] # Remove the width & height, change the border-left and border-right to use 5vw instead of 1vw.

(56) (styles.css) [cont'd] # Change the two #ggg to transparent.

(57) (styles.css) [cont'd] removing the margin & border-top properties.

(58) (styles.css) [cont'd] changing the border-bottom into:

```
border-bottom: 5vh solid var(--building-color2);
```

(59) (styles.css) [inside the :root { }

```
--window-color3: #d98cb3;
```

(60) (styles.css) specifying the direction of the gradient:

```
gradient-type(
  direction,
  color1,
  color2
);
```

```
background: repeating-linear-gradient(
  90deg,
  var(--building-color3),
  var(--building-color3),
  var(--window-color3) 15%
);
```

(61) (styles.css) removing the background-color property and value from .bb3

(62) (index.html) adding 3 div elements. → .bb4a, .bb4b, .bb4c.

(63) (styles.css) styling the width & height, for .bb4a - .bb4c.

(64) (styles.css) styling (again).

(65) (styles.css) adding a new class → .building-wrap { }

(66) (styles.css) moving properties only.

(67) (index.html) add the new building-wrap class to the .bb1 and .bb4 elements.

```
<div class="bb1 building-wrap"></div>
```

↓

bb4 also

(68) (styles.css) [within the :root { }] → --window-color4: #8cb3d9;

(69) (index.html) adding new 4 classes.

(70) (styles.css)

```
.bb4-window {
  width: 18%;
  height: 90%;
  background-color: var(--window-color4);
}
```

(71) (styles.css)

```
.window-wrap {
  display: flex;
  align-items: center;
  justify-content: space-evenly;
}
```

(72) (index.html) `<div class="bb4c window-wrap"></div>`

(77) (styles.css)

```
.fb1c {
  width: 100%;
  height: 80%;
  background: repeating-linear-gradient(90deg,
  var(--building-color4) 0%,
  var(--building-color4) 10%,
  transparent 10%,
  transparent 15%);
}
```

(78) (styles.css) adding multiple gradients

```
gradient1(
  colors
),
gradient2(
  colors
);
```

```
background: repeating-linear-gradient(
  90deg,
    var(--building-color4),
    var(--building-color4) 10%,
    transparent 10%,
    transparent 15%
),
  repeating-linear-gradient(
    var(--building-color4),
    var(--building-color4) 10%,
    var(--window-color4) 10%,
    var(--window-color4) 90%
);
```

(79) (styles.css)

```
.fb1a {
  border-bottom: 7vh solid var(--building-color4);
  border-left: 2vw solid transparent;
  border-right: 2vw solid transparent;
}
```

change this into 4vw.

(80)

(81) (index.html) adding 2 new div elements.

(styles.css) giving .fb2a .fb2b width * height.

(styles.css)

```
display: flex;
flex-wrap: wrap;
```

(styles.css)

```
.sky {
  background: radial-gradient(
  #ffcf33 0%, 20%,
  #ffff66 21%,
  #bbeeff 100%
  );
}
```

(styles.css) #fill the media query [see step 114] with the whole .sky { } [see step 112].

(styles.css)

```
:root {
  --building-color1: #000;
  --building-color2: #000;
  --building-color3: #000;
  --building-color4: #000;
}
```

.fb4a

(styles.css)

→ change into:

```
background: radial-gradient(
  circle closest-corner at 15% 15%,
```

(styles.css

```
@media (condition) {

}

@media (max-width: 1000px) {
}
```

(styles.css) [cont'd]

change --window-color1 until --window-color4 values into #777