LEARN CSS VARIABLES BY BUILDING A CITY SKYLINE

Introduction:

CSS variables help you organize your styles and reuse them.

In this course, you'll build a city skyline. You'll learn how to configure CSS variables so you can reuse them whenever you want.

Step 1:

Welcome to the CSS Variables Skyline project! Start by adding the `!DOCTYPE html` declaration at the top of the document so the browser knows what type of document it's reading.

Step 2:

Add opening and closing `html` tags below the `DOCTYPE` so you have a place to start putting some code. Be sure to set the language to English.

Step 3:

Next, add opening and closing `head` and `body` tags within the `html` element.

Step 4:

Within the `head`, nest a `meta` element with a `charset` of `UTF-8`, a `title` element with a title of `City Skyline`, and a `link` element that links your `styles.css` file.

Step 5:

In CSS, you can target everything with an asterisk. Add a border to everything by using the `*` selector, and giving it a `border` of `1px solid black`. This is a trick that helps visualize where elements are and their size. You will remove this later.

Step 6:

Also add a `box-sizing` of `border-box` to everything. This will make it so the border you added doesn't add any size to your elements.

Step 7:

You can see the `body` (it's the inner-most box on your page); the box around it is the `html` element. Make your `body` fill the whole viewport by giving it a `height` of `100vh`. Remove the default `margin` from the `body` by setting the `margin` to `0`. Finally, set the `overflow` property to `hidden` to hide any scroll bars that appear when something extends past the viewport.

Step 8:

Create a `div` element in the `body` with a class of `background-buildings`. This will be a container for a group of buildings.

Step 9:

Give your `.background-buildings` element a `width` and `height` of `100%` to make it the full width and height of its parent, the `body`.

Step 10:

Nest a `div` with a class of `bb1` in the background buildings container. Open your `styles.css` file, and give `.bb1` a `width` of `10%` and `height` of `70%`. "bb" stands for "background building", this will be your first building.

Step 11:

Nest four `div` elements in the `.bb1` container. Give them the classes `bb1a`, `bb1b`, `bb1c`, and `bb1d` in that order. This building will have four sections.

Step 12:

Give the parts of your building `width` and `height` properties with these values: `70%` and `10%` to `.bb1a`, `80%` and `10%` to `.bb1b`, `90%` and `10%` to `.bb1c`, and `100%` and `70%` to `.bb1d`. Remember that these percentages are relative to the parent and note that the heights will add up to 100% - vertically filling the container.

Step 13:

Center the parts of your building by turning the `.bb1` element into a flexbox parent. Use the `flex-direction` and `align-items` properties to center the children.

Step 14:

Now you have something that resembles a building. You are ready to create your first variable. Variable declarations begin with two 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`.

Step 15:

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.

Step 16:

Use the same variable as the `background-color` of the `.bb1b`, `.bb1c`, and `.bb1d` classes to fill in the rest of the building.

Step 17:

Change the value of your variable from `#999` to `#aa80ff` and you can see how it gets applied everywhere you used the variable. This is the main advantage of using variables, being able to quickly change many values in your stylesheet by just changing the value of a variable.

Step 18:

Your first building looks pretty good now. Nest three new `div` elements in the `.background-buildings` container and give them the classes of `bb2`, `bb3`, and `bb4` in that order. These will be three more buildings for the background.

Step 19:

Give the new buildings `width` and `height` properties of: `10%` and `50%` for `.bb2`, `10%` and `55%` for `.bb3`, and `11%` and `58%` for `.bb4`. You will be using almost all percent based units and some flexbox for this project, so everything will be completely responsive.

Step 20:

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 across the bottom of the element.

Step 21:

The buildings are too spaced out. Squeeze them together by adding two empty `div` elements to the top of the `.background-buildings` element, two more at the bottom of it, and one more in between `.bb3` and `.bb4`. These will be added as evenly-spaced elements across the container, effectively moving the buildings closer to the center.

Step 22:

Create a new variable below your `--building-color1` variable. Name your new variable `--building-color2` and give it a value of `#66cc99`. Then set it as the `background-color` of `.bb2`.

Step 23:

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

Step 24:

Create a new variable below the other ones named `--building-color3` and give it a value of `#cc6699`. Then use it as the `background-color` of the `.bb3` class and give it a fallback value of `pink`.

Step 25:

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.

Step 26:

Now that you've worked the bugs out and the buildings are the right colors, you can remove the fallback values in the two places they were used. Go ahead and do that now.

Step 27:

Create another variable named `--building-color4` and give it a value of `#538cc6`. Make sure it's in the `:root` selector this time. Then use it to fill in the last building.

Step 28:

The background buildings are starting to look pretty good. Create a new `div` below the `.background-buildings` element and give it a class of `foreground-buildings`. This will be another container for more buildings.

Step 29:

You want the `.foreground-buildings` container to sit directly on top of the `.background-buildings` element. Give it a `width` and `height` of `100%`, set the `position` to `absolute`, and the `top` to `0`. This will make it the same size as the body and move the start of it to the top left corner.

Step 30:

Nest six `div` elements within `.foreground-buildings` and give them the classes of `fb1` through `fb6` in that order. "fb" stands for "foreground building". These will be six more buildings for the foreground.

Step 31:

Give the six new elements these `width` and `height` values: `10%` and `60%` to `.fb1`, `10%` and `40%` to `.fb2`, `10%` and `35%` to `.fb3`, `8%` and `45%` to `.fb4`, `10%` and `33%` to `.fb5`, and `9%` and `38%` to `.fb6`.

Step 32:

Add the same `display`, `align-items`, and `justify-content` properties and values to `.foreground-buildings` that you used on `.background-buildings`. Again, this will use Flexbox to evenly space the buildings across the bottom of their container.

Step 33:

You should optimize your code. Move the position and top properties and values from .foreground-buildings to .background-buildings. Then select both .background-buildings and .foreground-buildings there, effectively applying those styles to both of the elements. You can use a comma (,) to separate selectors like this: selector1, selector2.

Step 34:

Now that you did that, you can delete the old .foreground-buildings declaration and all of its properties since they aren't needed anymore.

Step 35:

The skyline is coming together. Fill in the background-color property of the foreground buildings. Use your --building-color1 variable to fill in .fb3 and .fb4, --building-color2 for .fb5, --building-color3 for .fb2 and .fb6, and --building-color4 for .fb1.

Step 36:

Squeeze the buildings together again by adding two empty div elements within both the top and bottom of the .foreground-buildings element, and one more in between .fb2 and .fb3.

Step 37:

Move the position of .fb4 relative to where it is now by adding a position of relative and left of 10% to it. Do the same for .fb5 but use right instead of left. This will cover up the remaining white space in between the buildings.

Step 38:

Your code is starting to get quite long. Add a comment above the `.fb1` class that says `FOREGROUND BUILDINGS - "fb" stands for "foreground building"` to help people understand your code. Above the `.bb1` class add another comment that says `BACKGROUND BUILDINGS - "bb" stands for "background building"`. If you don't remember, comments in CSS look like this: `/* Comment here */`.

Step 39:

Create a new variable in `:root` called `--window-color1` and give it a value of `black`. This will be a secondary color for the purple buildings.

Step 40:

Gradients in CSS are a way to transition between colors across the distance of an element. They are applied to the `background` property and the syntax looks like this:

Example Code:

```
gradient-type(

  color1,

  color2

);
```

In the example, `color1` is solid at the top, `color2` is solid at the bottom, and in between it transitions evenly from one to the next. In `.bb1a`, add a `background` property below the `background-color` property. Set it as a gradient of type `linear-gradient` that uses

`--building-color1` as the first color and `--window-color1` as the second.

Step 41:

You want to add the same gradient to the next two sections. Instead of doing that, create a new class selector called `bb1-window`, and move the `height` and `background` properties and values from `.bb1a` to the new class selector.

Step 42:

Add the new `bb1-window` class to the `.bb1a`, `.bb1b`, and `.bb1c` elements. This will apply the gradient to them.

Step 43:

You don't need the `height` or `background-color` properties in `.bb1a`, `.bb1b` or `.bb1c` anymore, so go ahead and remove them.

Step 44:

Gradients can use as many colors as you want like this:

Example Code:

```
gradient-type(
  color1,
  color2,
  color3
);
```

Add a linear-gradient to .bb1d with orange as the first color, --building-color1 as the second, and --window-color1 as the third. Remember to use the gradient on the background property.

Step 45:

It's a little hidden behind the foreground buildings, but you can see the three color gradient there. Since you are using that now, remove the background-color property from .bb1d.

Step 46:

You can specify where you want a gradient transition to complete by adding it to the color like this:

Example Code:

```
gradient-type(
  color1,
  color2 20%,
  color3
);
```

Here, it will transition from color1 to color2 between 0% and 20% of the element and then transition to color3 for the rest. Add 80% to the --building-color1 color of the .bb1d gradient so you can see it in action.

Step 47:

Remove orange from the .bb1d gradient and change the 80% to 50%. This will make --building-color1 solid for the top half, and then transition to --window-color1 for the bottom half.

Step 48:

Nest two new div elements within .bb2, give them the classes of bb2a and bb2b, in that order. These will be two sections for this building.

Step 49:

Give .bb2b a width and height of 100% to make it fill the building container. You will add something on the top a little later.

Step 50:

Create a new variable in :root named window-color2 with a value of #8cd9b3. This will be used as the secondary color for this building.

Step 51:

Gradient transitions often gradually change from one color to another. When a more abrupt change is required, the transition can be made with a hard stop like this:

Example Code:

```
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%`.

Step 52:

You can see the hard color change at the top of the section. Change the gradient type from `linear-gradient` to `repeating-linear-gradient` for this section. This will make the four colors of your gradient repeat until it gets to the bottom of the element; giving you some stripes, and saving you from having to add a bunch of elements to create them.

Step 53:

In the next few steps, you are going to use some tricks with CSS borders to turn the `.bb2a` section into a triangle at the top of the building. First, remove the `background-color` from `.bb2` since you don't need it anymore.

Step 54:

Create and add the following properties to `.bb2a`:

Example Code:

```
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;`

After you add these, you can see how a thick border on an element gives you some angles where two sides meet. You are going to use that bottom border as the top of the building.

Step 55:

Next, remove the `width` and `height` from `.bb2a`, and change the `border-left` and `border-right` to use `5vw` instead of `1vw`. The element will now have zero size and the borders will come together in the middle.

Step 56:

Next, change the two `#999` of `.bb2a` to `transparent`. This will make the left and right borders invisible.

Step 57:

Remove the `margin` and `border-top` properties and values from `.bb2a` to turn it into a triangle for the top of the building.

Step 58:

Finally, on the `border-bottom` property of `.bb2a`, change the `1vw` to `5vh` and change the `#000` color to your `--building-color2` variable. There you go, now it looks good! At any time throughout this project, you can comment out or remove the `border` property you added to everything

at the beginning to see what the buildings will look like when that gets removed at the end.

Step 59:

On to the next building! Create a new variable called `--window-color3` in `:root` and give it a value of `#d98cb3`. This will be the secondary color for the pink buildings.

Step 60:

So far, all the gradients you created have gone from top to bottom, that's the default direction. You can specify another direction by adding it before your colors like this:

Example Code:

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

Fill in `.bb3` with a `repeating-linear-gradient`. Use `90deg` for the direction, your `building-color3` for the first two colors, and `window-color3` at `15%` for the third.

When you don't specify a distance for a color, it will use the values that make sense. In this case, the first two colors will default to `0%` and `7.5%` because it starts at `0%`, and `7.5%` is half of the `15%`, so you do not need to set them.

Step 61:

Remove the background-color property and value from .bb3 since you are using the gradient as the background now.

Step 62:

The next building will have three sections. Nest three div elements within .bb4. Give them the classes of bb4a, bb4b and bb4c in that order.

Step 63:

Give the new div elements these width and height values: 3% and 10% to .bb4a, 80% and 5% to .bb4b, and 100% and 85% to .bb4c.

Step 64:

Remove the background-color property and value from .bb4, and add it to the three new sections .bb4a, .bb4b, and .bb4c, so only the sections are filled.

Step 65:

You want .bb4 to share the properties of .bb1 that center the sections. Instead of duplicating that code, create a new class above the background building comment called building-wrap. Leave it empty for now; this class will be used in a few places to save you some coding.

Step 66:

Move the display, flex-direction, and align-items properties and values from .bb1 to the new building-wrap class.

Step 67:

Add the new building-wrap class to the .bb1 and .bb4 elements. This will apply the centering properties to the buildings that need it.

Step 68:

Create a new variable called --window-color4 in :root and give it a value of #8cb3d9. This will be the secondary color for the last background building.

Step 69:

Nest four new div elements within .bb4c, give them all the class of bb4-window. These will be windows for this building.

Step 70:

Give the bb4-window class a width of 18%, a height of 90%, and add your --window-color4 variable as the background-color.

Step 71:

The windows are stacked on top of each other at the left of the section, behind the purple building. Add a new class below .building-wrap called window-wrap. Make .window-wrap a flexbox container, and use the align-items and justify-content properties to center its child elements vertically and evenly space them in their parent, respectively.

Step 72:

Add the new window-wrap class to the .bb4c element.

Step 73:

Looks good! On to the foreground buildings! Turn the .fb1 building into three sections by nesting three new div elements within it. Give them the classes of fb1a, fb1b and fb1c, in that order.

Step 74:

Give .fb1b a width of 60% and height of 10%, and .fb1c a width of 100% and height of 80%.

Step 75:

Add the building-wrap class to the .fb1 element to center the sections.

Step 76:

Move the background-color property and value from .fb1 to .fb1b.

Step 77:

Don't worry about the space at the bottom, everything will get moved down later when you add some height to the element at the top of the building.

Add a repeating-linear-gradient to .fb1c with a 90deg angle, your --building-color4 from 0% to 10% and transparent from 10% to 15%.

Step 78:

You can add multiple gradients to an element by separating them with a comma (,) like this:

Example Code:

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

Add a repeating-linear-gradient to .fb1c below the one that's there; use your --building-color4 from 0% to 10% and --window-color4 from 10% and 90%. This will fill in behind the gradient you added last.

Step 79:

You're going to use some more border tricks for the top section. Add a border-bottom with a value of 7vh solid var(--building-color4) to .fb1a. This will put a 7vh height border on the bottom. But since the element has zero size, it only shows up as a 2px wide line from the 1px border that is on all the elements.

Step 80:

When you increase the size of the left and right borders, the border on the bottom will expand to be the width of the combined left and right border widths. Add `2vw solid transparent` as the value of the `border-left` and `border-right` properties of `.fb1a`. They will be invisible, but it will make the border on the bottom `4vw` wide.

Step 81:

On to the next building! Nest two `div` elements within `.fb2` and give them classes of `fb2a` and `fb2b`, in that order.

Step 82:

Give `.fb2a` a `width` of `100%` and `.fb2b` a `width` of `100%` and `height` of `75%`.

Step 83:

Nest three `div` elements within `.fb2b` and give them a class of `fb2-window`. These will be windows for this section of the building.

Step 84:

Add your `window-wrap` class to `.fb2b` to position the new window elements.

Step 85:

Give the `.fb2-window` elements a `width` of `22%`, `height` of `100%`, and a `background-color` of your `--window-color3` variable.

Step 86:

Move the background-color property and value from .fb2 to .fb2b to just color the section and not the container.

Step 87:

For .fb2a, add a border-bottom of 10vh solid var(--building-color3), and a border-left and border-right of 1vw solid transparent. This time the border trick will create a trapezoid shape.

Step 88:

For the next building, nest four div elements within .fb3 with classes of fb3a, fb3b, fb3a again, and fb3b again, in that order. This building will have four sections, and the top two will be almost the same as the bottom two.

Step 89:

Give the .fb3a element a width of 80% and height of 15%. Then give the .fb3b element a width of 100% and height of 35%.

Step 90:

Remove the background-color property and value from .fb3, and add them to .fb3a and .fb3b.

Step 91:

Add your building-wrap class to the .fb3 element to center the sections.

Step 92:

Nest three new `div` elements in the first `.fb3a` element. Give them each a class of `fb3-window`. These will be windows for this section.

Step 93:

Give the `.fb3-window` elements a `width` of `25%`, a `height` of `80%`, and use your `--window-color1` variable as the `background-color` value.

Step 94:

Add your `window-wrap` class to the `.fb3a` element to center and space the windows.

Step 95:

With CSS variables you can change values without searching everywhere in the stylesheet. Change the `--window-color1` value to `#bb99ff`.

Step 96:

Only three more buildings to go. Nest two new `div` elements within the `.fb4` element and give them the classes of `fb4a` and `fb4b`, in that order. Remember that you sort of flipped the location of `.fb4` and `.fb5`, so it's the rightmost purple building you are working on now.

Step 97:

Give `.fb4b` a `width` of `100%` and `height` of `89%`.

Step 98:

Add your `--building-color1` variable as value of the `background-color` property of `.fb4b`. Then, remove the `background-color` from `.fb4`.

Step 99:

Nest six `div` elements within `.fb4b` and give them all a class of `fb4-window`.

Step 100:

Give the `.fb4-window` elements a `width` of `30%`, `height` of `10%`, and `border-radius` of `50%`. These will make some circular windows for this building.

Step 101:

Fill in the windows with your secondary color for this building. Also add a `margin` of `10%` to give the windows some space.

Step 102:

The windows are stacked on top of each other on the rightmost purple building. Turn the building into a flexbox parent, and use the `flex-wrap` property to put the windows side by side, and push them down to a new row when they don't fit.

Step 103:

This building is going to have another triangle on top. Give the top section a `border-top` of `5vh solid transparent`, and a `border-left` that is `8vw`, `solid`, and uses your building color variable as the color.

Step 104:

On to the next building! It's the green one in the foreground. Give it a `repeating-linear-gradient` with your building color from `0%` to `5%`, and `transparent` from `5%` to `10%`.

Step 105:

Add another `repeating-linear-gradient` below the one you just added. Give it a `90deg` direction, use your building color from `0%` to `12%` and window color `12%` to `44%`. This will make a bunch of rectangle windows.

Step 106:

You don't need the `background-color` for this building anymore so you can remove that property.

Step 107:

Finally! You made it to the last building! Add a repeating gradient to it with a `90deg` direction. Use the building color from `0%` to `10%` and `transparent` from `10%` to `30%`.

Step 108:

Add another repeating gradient to this building; make it the same as the one you just added, except don't add the `90deg` direction and use your window color instead of the two `transparent` colors.

Step 109:

You can remove the background-color for this building now, since it isn't needed.

Step 110:

Okay, the buildings are done. Go back to the * selector and remove the border you applied to everything at the beginning and the buildings will come together.

Step 111:

Add sky as a second class to the .background-buildings element. You are going to make a background for the skyline.

Step 112:

Give the sky class a radial-gradient. Use #ffcf33 from 0% to 20%, #ffff66 at 21%, and #bbeeff at 100%. This will add circular gradient to the background that will be your sun.

Step 113:

At the top of the sky gradient color list, where you would put a direction for the gradient; add circle closest-corner at 15% 15%,. This will move the start of the gradient to 15% from the top and left. It will make it end at the closest-corner and it will maintain a circle shape. These are some keywords built into gradients to describe how it behaves.

Step 114:

A media query can be used to change styles based on certain conditions, and they look like this:

Example Code:

```
@media (condition) {

}
```

Add an empty media query at the bottom of your stylesheet with a condition of `max-width: 1000px`. Styles added in here will take effect when the document size is 1000px wide or less.

Step 115:

Copy and paste your whole `sky` class along with all of its properties and values into the media query. You are going to make another color scheme for the skyline that changes it from day to night.

Note: You are going to need to scroll past the editable region to copy the class.

Step 116:

In the `sky` class of the media query, change the two `#ffcf33` color values to `#ccc`, the `#ffff66` to `#445`, and the `#bbeeff` to `#223`. Then you can resize your window to see the background change colors.

Step 117:

Add a `:root` selector to the top of your media query. Then redefine all four of the `--building-color` variables to use the value `#000` there.

Step 118:

Lastly, in the `:root` selector of the media query, redefine all four of the `--window-color` variables to use `#777`. When you're done, resize the window and watch it go from day to night.

Variables are primarily used with colors, and that's how you used them here. But they can be given any value and used on any property. Your project looks great!