Content

- **Google fonts**
- **Attribute Selector [type=' ']**
- **Different units like em, px etc**
- **! important**
- **s and hr tag**
- **rgba()**
- **box-shadow**
- **opacity?**
- **text-transform**
- **line-height**
- **hover**
- **position : relative, absolute, fixed**
- **float: left or right**
- **z-index**
- **margin: auto**
- **hsl()**
- **linear-gradient**
- **url()**
- **scale(1.5);**
- **skewX & Y**
- **animation**

Basic CSS: Import a Google FontPassed

In addition to specifying common fonts that are found on most operating systems, we can also specify non-standard, custom web fonts for use on our website. There are many sources for web fonts on the Internet. For this example we will focus on the Google Fonts library.

Google Fonts is a free library of web fonts that you can use in your CSS by referencing the font's URL.

So, let's go ahead and import and apply a Google font (note that if Google is blocked in your country, you will need to skip this challenge).

To import a Google Font, you can copy the font(s) URL from the Google Fonts library and then paste it in your HTML. For this challenge, we'll import the Lobster font. To do this, copy the following code snippet and paste it into the top of your code editor (before the opening style element):

<link href="https://fonts.googleapis.com/css?family=Lobster" rel="stylesheet" type="text/css">

Now you can use the Lobster font in your CSS by using Lobster as the FAMILY_NAME as in the following example:

font-family: FAMILY_NAME, GENERIC_NAME;.

The GENERIC_NAME is optional, and is a fallback font in case the other specified font is not available. This is covered in the next challenge.

Family names are case-sensitive and need to be wrapped in quotes if there is a space in the name. For example, you need quotes to use the "Open Sans" font, but not to use the Lobster font.

==========

Basic CSS: Use Attribute Selectors to Style ElementsPassed

You have been adding id or class attributes to elements that you wish to specifically style. These are known as ID and class selectors. There are other CSS Selectors you can use to select custom groups of elements to style.

Let's bring out CatPhotoApp again to practice using CSS Selectors.

For this challenge, you will use the [attr=value] attribute selector to style the checkboxes in CatPhotoApp. This selector matches and styles elements with a specific attribute value. For example, the below code changes the margins of all elements with the attribute type and a corresponding value of radio:

```
[type='radio'] {
  margin: 20px 0px 20px 0px;
}
```

==========

Basic CSS: Understand Absolute versus Relative UnitsPassed

The last several challenges all set an element's margin or padding with pixels (px). Pixels are a type of length unit, which is what tells the browser how to size or space an item. In addition to px, CSS has a number of different length unit options that you can use.

The two main types of length units are absolute and relative. Absolute units tie to physical units of length. For example, in and mm refer to inches and millimeters, respectively. Absolute length units approximate the actual measurement on a screen, but there are some differences depending on a screen's resolution.

Relative units, such as em or rem, are relative to another length value. For example, em is based on the size of an element's font. If you use it to set the font-size property itself, it's relative to the parent's font-size.

Note: There are several relative unit options that are tied to the size of the viewport. They are covered in the Responsive Web Design Principles section.

==========

Basic CSS: Override All Other Styles by using Important

Yay! We just proved that inline styles will override all the CSS declarations in your style element.

But wait. There's one last way to override CSS. This is the most powerful method of all. But before we do it, let's talk about why you would ever want to override CSS.

In many situations, you will use CSS libraries. These may accidentally override your own CSS. So when you absolutely need to be sure that an element has specific CSS, you can use !important

Let's go all the way back to our pink-text class declaration. Remember that our pink-text class was overridden by subsequent **class declarations**, *id declarations*, and inline styles.

Let's add the keyword !important to your pink-text element's color declaration to make 100% sure that your h1 element will be pink.

An example of how to do this is:

color: red !important;

==========

Basic CSS: Use CSS Variables to change several elements at oncePassed

CSS Variables are a powerful way to change many CSS style properties at once by changing only one value.

Follow the instructions below to see how changing just three values can change the styling of many elements.

```
<style>

 .penguin {

   /* Only change code below this line */
   --penguin-skin: black;
   --penguin-belly: gray;
   --penguin-beak: yellow;
   /* Only change code above this line */

   position: relative;
   margin: auto;
   display: block;
   margin-top: 5%;
   width: 300px;
   height: 300px;
 }

 .penguin-top {
   top: 10%;
   left: 25%;
   background: var(--penguin-skin, gray);
   width: 50%;
   height: 45%;
   border-radius: 70% 70% 60% 60%;
 }

 .penguin-bottom {
   top: 40%;
   left: 23.5%;
   background: var(--penguin-skin, gray);
   width: 53%;
   height: 45%;
   border-radius: 70% 70% 100% 100%;
 }

</style>
```

==========

Applied Visual Design: Use the s Tag to Strikethrough TextPassed

To strikethrough text, which is when a horizontal line cuts across the characters, you can use the s tag. It shows that a section of text is no longer valid. With the s tag, the browser applies the CSS of text-decoration: line-through; to the element.

Applied Visual Design: Create a Horizontal Line Using the hr ElementPassed

You can use the hr tag to add a horizontal line across the width of its containing element. This can be used to define a change in topic or to visually separate groups of content.

==========

Applied Visual Design: Adjust the background-color Property of Text

Instead of adjusting your overall background or the color of the text to make the foreground easily readable, you can add a background-color to the element holding the text you want to emphasize. This challenge uses rgba() instead of hex codes or normal rgb().

rgba stands for:

r = red

g = green

b = blue

a = alpha/level of opacity

The RGB values can range from 0 to 255. The alpha value can range from 1, which is fully opaque or a solid color, to 0, which is fully transparent or clear. rgba() is great to use in this case, as it allows you to adjust the opacity. This means you don't have to completely block out the background.

You'll use background-color: rgba(45, 45, 45, 0.1) for this challenge. It produces a dark gray color that is nearly transparent given the low opacity value of 0.1.

==========

Applied Visual Design: Add a box-shadow to a Card-like Element

The box-shadow property applies one or more shadows to an element.

The box-shadow property takes values for

offset-x (how far to push the shadow horizontally from the element),

offset-y (how far to push the shadow vertically from the element),

blur-radius,

spread-radius and

color, in that order.

The blur-radius and spread-radius values are optional.

Multiple box-shadows can be created by using commas to separate properties of each box-shadow element.

Here's an example of the CSS to create multiple shadows with some blur, at mostly-transparent black colors:

box-shadow: 0 10px 20px rgba(0,0,0,0.19), 0 6px 6px rgba(0,0,0,0.23);

box-shadow: 25px 10px 10px 10px green; ***//in one line***

==========

Applied Visual Design: Decrease the Opacity of an Element

The opacity property in CSS is used to adjust the opacity, or conversely, the transparency for an item.

A value of 1 is opaque, which isn't transparent at all.

A value of 0.5 is half see-through.

A value of 0 is completely transparent.

The value given will apply to the entire element, whether that's an image with some transparency, or the foreground and background colors for a block of text

==========

Applied Visual Design: Use the text-transform Property to Make Text Uppercase

The text-transform property in CSS is used to change the appearance of text. It's a convenient way to make sure text on a webpage appears consistently, without having to change the text content of the actual HTML elements.

The following table shows how the different text-transformvalues change the example text "Transform me".

Value **Result**

lowercase	"transform me"
uppercase	"TRANSFORM ME"
capitalize	"Transform Me"
initial	Use the default value
inherit	Use the text-transform value from the parent element
none	Default: Use the original text

==========

Applied Visual Design: Set the line-height of Paragraphs

CSS offers the line-height property to change the height of each line in a block of text. As the name suggests, it changes the amount of vertical space that each line of text gets.

line-height: 20px;

also, see font-size and font-weight property.

==========

Applied Visual Design: Adjust the Hover State of an Anchor TagPassed

This challenge will touch on the usage of pseudo-classes. A pseudo-class is a keyword that can be added to selectors, in order to select a specific state of the element.

For example, the styling of an anchor tag can be changed for its hover state using the :hover pseudo-class selector. Here's the CSS to change the color of the anchor tag to red during its hover state:

```
a:hover {
  color: red;
}
```

==========

Applied Visual Design: Change an Element's Relative Position

CSS treats each HTML element as its own box, which is usually referred to as the CSS Box Model. Block-level items automatically start on a new line (think headings, paragraphs, and divs) while inline items sit within surrounding content (like images or spans). The default layout of elements in this way is called the normal flow of a document, but CSS offers the position property to override it.

When the position of an element is set to relative, it allows you to specify how CSS should move it relative to its current position in the normal flow of the page. It pairs with the CSS offset properties of left or right, and top or bottom. These say how many pixels, percentages, or ems to move the item away from where it is normally positioned. The following example moves the paragraph 10 pixels away from the bottom:

```
p {
  position: relative;
  bottom: 10px;
}
```

Changing an element's position to relative does not remove it from the normal flow - other elements around it still behave as if that item were in its default position. Note: Positioning gives you a lot of flexibility and power over the visual layout of a page. It's good to remember that no matter the position of elements, the underlying HTML markup should be organized and make sense when read from top to bottom. This is how users with visual impairments (who rely on assistive devices like screen readers) access your content.

The CSS offsets of top or bottom, and left or right tell the browser how far to offset an item relative to where it would sit in the normal flow of the document. You're offsetting an element away from a given spot, which moves the element away from the referenced side (effectively, the opposite direction). As you saw in the last challenge, using the top offset moved the h2 downwards. Likewise, using a left offset moves an item to the right.

Applied Visual Design: Lock an Element to its Parent with Absolute PositioningPassed

The next option for the CSS position property is absolute, which locks the element in place relative to its parent container. Unlike the relative position, this removes the element from the normal flow of the document, so surrounding items ignore it. The CSS offset properties (top or bottom and left or right) are used to adjust the position.

One nuance with absolute positioning is that it will be locked relative to its closest positioned ancestor. If you forget to add a position rule to the parent item, (this is typically done using position: relative;), the browser will keep looking up the chain and ultimately default to the body tag.

Applied Visual Design: Lock an Element to the Browser Window with Fixed PositioningPassed

The next layout scheme that CSS offers is the fixed position, which is a type of absolute positioning that locks an element relative to the browser window. Similar to absolute positioning, it's used with the CSS offset properties and also removes the element from the normal flow of the document. Other items no longer "realize" where it is positioned, which may require some layout adjustments elsewhere.

One key difference between the fixed and absolute positions is that an element with a fixed position won't move when the user scrolls.

==========

Applied Visual Design: Push Elements Left or Right with the float PropertyPassed

The next positioning tool does not actually use position, but sets the float property of an element. Floating elements are removed from the normal flow of a document and pushed to either the left or right of their containing parent element. It's commonly used with the width property to specify how much horizontal space the floated element requires.

==========

Applied Visual Design: Change the Position of Overlapping Elements with the z-index Property

When elements are positioned to overlap (i.e. using position: absolute | relative | fixed | sticky), the element coming later in the HTML markup will, by default, appear on the top of the other elements. However, the z-index property can specify the order of how elements are stacked on top of one another. It must be an integer (i.e. a whole number and not a decimal), and higher values for the z-index property of an element move it higher in the stack than those with lower values.

Add a z-index property to the element with the class name of first (the red rectangle) and set it to a value of 2 so it covers the other element (blue rectangle).

==========

Applied Visual Design: Center an Element Horizontally Using the margin Property

Another positioning technique is to center a block element horizontally. One way to do this is to set its

margin to a value of auto.

This method works for images, too. Images are inline elements by default, but can be changed to block elements when you set the display property to block.

==========

Applied Visual Design: Adjust the Hue of a Color

Colors have several characteristics including hue, saturation, and lightness. CSS3 introduced the hsl() property as an alternative way to pick a color by directly stating these characteristics.

Hue is what people generally think of as 'color'. If you picture a spectrum of colors starting with red on the left, moving through green in the middle, and blue on right, the hue is where a color fits along this line. In hsl(), hue uses a color wheel concept instead of the spectrum, where the angle of the color on the circle is given as a value between 0 and 360.

Saturation is the amount of gray in a color. A fully saturated color has no gray in it, and a minimally saturated color is almost completely gray. This is given as a percentage with 100% being fully saturated.

Lightness is the amount of white or black in a color. A percentage is given ranging from 0% (black) to 100% (white), where 50% is the normal color.

Here are a few examples of using hsl() with fully-saturated, normal lightness colors:

Color	HSL
red	hsl(0, 100%, 50%)
yellow	hsl(60, 100%, 50%)
green	hsl(120, 100%, 50%)
cyan	hsl(180, 100%, 50%)
blue	hsl(240, 100%, 50%)
magenta	hsl(300, 100%, 50%)

==========

Applied Visual Design: Create a Gradual CSS Linear Gradient

Applying a color on HTML elements is not limited to one flat hue. CSS provides the ability to use color transitions, otherwise known as gradients, on elements. This is accessed through the background property's linear-gradient() function. Here is the general syntax:

background: linear-gradient(gradient_direction, color 1, color 2, color 3, ...);

The first argument specifies the direction from which color transition starts - it can be stated as a degree, where 90deg makes a vertical gradient and 45deg is angled like a backslash. The following arguments specify the order of colors used in the gradient.

Example:

background: linear-gradient(90deg, red, yellow, rgb(204, 204, 255));

==========

Applied Visual Design: Create Texture by Adding a Subtle Pattern as a Background Image

One way to add texture and interest to a background and have it stand out more is to add a subtle pattern. The key is balance, as you don't want the background to stand out too much, and take away from the foreground. The background property supports the **url()** function in order to link to an image of the chosen texture or pattern. The link address is wrapped in quotes inside the parentheses.

==========

Applied Visual Design: Use the CSS Transform scale Property to Scale an Element on HoverPassed

The transform property has a variety of functions that let you scale, move, rotate, skew, etc., your elements. When used with pseudo-classes such as :hover that specify a certain state of an element, the transform property can easily add interactivity to your elements.

Here's an example to scale the paragraph elements to 2.1 times their original size when a user hovers over them:

```
p:hover {
  transform: scale(2.1);
}
```

Note: Applying a transform to a div element will also affect any child elements contained in the div.

Applied Visual Design: Use the CSS Transform Property skewX to Skew an Element Along the X-Axis

The next function of the transform property is skewX(), which skews the selected element along its X (horizontal) axis by a given degree.

The following code skews the paragraph element by -32 degrees along the X-axis.

```
p {
```

```
 transform: skewX(-32deg);
}
```

==========

Applied Visual Design: Learn How the CSS @keyframes and animation Properties Work

To animate an element, you need to know about the animation properties and the @keyframes rule. The animation properties control how the animation should behave and the @keyframes rule controls what happens during that animation. There are eight animation properties in total. This challenge will keep it simple and cover the two most important ones first:

animation-name sets the name of the animation, which is later used by @keyframes to tell CSS which rules go with which animations.

animation-duration sets the length of time for the animation.

@keyframes is how to specify exactly what happens within the animation over the duration. This is done by giving CSS properties for specific "frames" during the animation, with percentages ranging from 0% to 100%. If you compare this to a movie, the CSS properties for 0% is how the element displays in the opening scene. The CSS properties for 100% is how the element appears at the end, right before the credits roll. Then CSS applies the magic to transition the element over the given duration to act out the scene. Here's an example to illustrate the usage of @keyframes and the animation properties:

```
#anim {
 animation-name: colorful;
 animation-duration: 3s;
}
@keyframes colorful {
 0% {
  background-color: blue;
 }
 100% {
  background-color: yellow;
 }
```

```
}
```

For the element with the anim id, the code snippet above sets the animation-name to colorful and sets the animation-duration to 3 seconds. Then the @keyframes rule links to the animation properties with the name colorful. It sets the color to blue at the beginning of the animation (0%) which will transition to yellow by the end of the animation (100%). You aren't limited to only beginning-end transitions, you can set properties for the element for any percentage between 0% and 100%.

Applied Visual Design: Use CSS Animation to Change the Hover State of a Button

You can use CSS @keyframes to change the color of a button in its hover state.

Here's an example of changing the width of an image on hover:

```
<style>
 img:hover {
  animation-name: width;
  animation-duration: 500ms;
 }
 @keyframes width {
  100% {
   width: 40px;
  }
 }
</style>
<img src="https://bit.ly/smallgooglelogo" alt="Google's Logo" />
```

Note that ms stands for milliseconds, where 1000ms is equal to 1s.

Use CSS @keyframes to change the background-color of the button element so it becomes #4791d0 when a user hovers over it. The @keyframes rule should only have an entry for 100%.

Applied Visual Design: Modify Fill Mode of an Animation

That's great, but it doesn't work right yet. Notice how the animation resets after 500ms has passed, causing the button to revert back to the original color. You want the button to stay highlighted.

This can be done by setting the animation-fill-mode property to forwards. The animation-fill-mode specifies the style applied to an element when the animation has finished. You can set it like so:

animation-fill-mode: forwards;

Applied Visual Design: Create Movement Using CSS Animation

When elements have a specified position, such as fixed or relative, the CSS offset properties right, left, top, and bottom can be used in animation rules to create movement.

As shown in the example below, you can push the item downwards then upwards by setting the top property of the 50% keyframe to 50px, but having it set to 0px for the first (0%) and the last (100%) keyframe.

```
@keyframes rainbow {
  0% {
    background-color: blue;
    top: 0px;
  }
  50% {
    background-color: green;
    top: 50px;
  }
  100% {
    background-color: yellow;
    top: 0px;
  }
}
```

Add a horizontal motion to the div animation. Using the left offset property, add to the @keyframes rule so rainbow starts at 0 pixels at 0%, moves to 25 pixels at 50%, and ends at -25 pixels at 100%. Don't replace the top property in the editor - the animation should have both vertical and horizontal motion.

Applied Visual Design: Animate Elements Continually Using an Infinite Animation Count

The previous challenges covered how to use some of the animation properties and the @keyframes rule. Another animation property is the animation-iteration-count, which allows you to control how many times you would like to loop through the animation. Here's an example:

animation-iteration-count: 3;

In this case the animation will stop after running 3 times, but it's possible to make the animation run continuously by setting that value to infinite.

Applied Visual Design: Change Animation Timing with Keywords

In CSS animations, the animation-timing-function property controls how quickly an animated element changes over the duration of the animation. If the animation is a car moving from point A to point B in a given time (your animation-duration), the **animation-timing-function** says how the car accelerates and decelerates over the course of the drive.

There are a number of predefined keywords available for popular options. For example, the default value is ***ease***, which starts slow, speeds up in the middle, and then slows down again in the end. Other options include ***ease-out***, which is quick in the beginning then slows down, ease-in, which is slow in the beginning, then speeds up at the end, or ***linear***, which applies a constant animation speed throughout.

For the elements with id of ball1 and ball2, add an animation-timing-function property to each, and set #ball1 to linear, and #ball2 to ease-out. Notice the difference between how the elements move during the animation but end together, since they share the same animation-duration of 2 seconds.

Applied Visual Design: Learn How Bezier Curves Work

The last challenge introduced the animation-timing-function property and a few keywords that change the speed of an animation over its duration. CSS offers an option other than keywords that provides even finer control over how the animation plays out, through the use of Bezier curves.

In CSS animations, Bezier curves are used with the cubic-bezier function. The shape of the curve represents how the animation plays out. The curve lives on a 1 by 1 coordinate system. The X-axis of this coordinate system is the duration of the animation (think of it as a time scale), and the Y-axis is the change in the animation.

The cubic-bezier function consists of four main points that sit on this 1 by 1 grid: p0, p1, p2, and p3. p0 and p3 are set for you - they are the beginning and end points which are always located respectively at the origin (0, 0) and (1, 1). You set the x and y values for the other two points, and where you place them in the grid dictates the shape of the curve for the animation to follow. This is done in CSS by declaring the x and y values of the p1 and p2 "anchor" points in the form: (x1, y1, x2, y2). Pulling it all together, here's an example of a Bezier curve in CSS code:

animation-timing-function: cubic-bezier(0.25, 0.25, 0.75, 0.75);

In the example above, the x and y values are equivalent for each point (x1 = 0.25 = y1 and x2 = 0.75 = y2), which if you remember from geometry class, results in a line that extends from the origin to point (1, 1). This animation is a linear change of an element during the length of an animation, and is the same as using the linear keyword. In other words, it changes at a constant speed.