1. Metadata elements of html are link, meta, title and style ------> These tags go inside head tag of html.
2. Cascading Style Sheets (CSS) tells the browser how to display the texts.
3. CSS allows us to control

* color
* fonts
* positioning
* spacing
* sizing
* decorations
* transitions

1. Three main ways to apply CSS styling

* Inline style sheets directly to HTML elements.
* Applying CSS rules with in style tags of HTML.
* External Style Sheets.

1. The idea behind CSS is that you can use a selector to target an HTML element in the DOM (Document Object Model) and then apply a variety of attributes to that element to change the way it is displayed on the page.
2. In your CSS style element, class names start with a period. In your HTML elements' class attribute, the class name does not include the period.
3. <style>

.red-text {

color: red;

}

</style>

<h2 class="red-text">CatPhotoApp</h2>

1. An id also has a higher specificity (importance) than a class so if both are applied to the same element and have conflicting styles, the styles of the id will be applied.
2. Note that inside your style element, you always reference classes by putting a .in front of their names. You always reference ids by putting a # in front of their names.
3. Three important properties control the space that surrounds each HTML element: padding, margin, and border.
4. An element's padding controls the amount of space between the element's content and its border.
5. If you set an element's margin to a negative value, the element will grow larger.
6. The two main types of length units are absolute and relative.
7. **Margin** is the outer space of an element, while **padding** is the inner space of an element. In other words, **margin** is the space outside of an element's border, while **padding** is the space inside of its border.
8. To declare a variable in CSS the notation is –variable name:---

Example: --penguin-skin: white;

1. In web development, HTML gives structure and semantics to a page's content, and CSS controls the layout and appearance of it.
2. Text is often a large part of web content. CSS has several options for how to align it with the text-align property.
3. text-align: justify; causes all lines of text except the last line to meet the left and right edges of the line box.
4. text-align: center; centers the text
5. text-align: right; right-aligns the text
6. And text-align: left; (the default) left-aligns the text.
7. To make text bold, you can use the strong tag.
8. To underline text, you can use the u tag.
9. To emphasize text, you can use the em tag. This displays text as italicized.
10. We can use the hr tag to add a horizontal line across the width of its containing element.

Hr tag is a self-closing tag so need of closing manually

<hr>.

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

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

1. The text-transformproperty in CSS is used to change the appearance of text.

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

1. The font-weight property sets how thick or thin characters are in a section of text.
2. 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.
3. 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;  
}

1. CSS treats each HTML element as its own box, which is usually referred to as the CSS Box Model.
2. 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;  
}

1. 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.
2. 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 from the absolute position is that the element won't move when the user scrolls.
3. 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.
4. When elements are positioned to overlap, 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.
5. 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.
6. When two colors are opposite each other on the wheel, they are called complementary colors. They have the characteristic that if they are combined, they "cancel" each other out and create a gray color. However, when placed side-by-side, these colors appear more vibrant and produce a strong visual contrast.

Some examples of complementary colors with their hex codes are:

red (#FF0000) and cyan (#00FFFF)  
green (#00FF00) and magenta (#FF00FF)  
blue (#0000FF) and yellow (#FFFF00)

1. Red (R), green (G), and blue (B) are called primary colors. Mixing two primary colors creates the secondary colors cyan (G + B), magenta (R + B) and yellow (R + G). These secondary colors happen to be the complement to the primary color not used in their creation, and are opposite to that primary color on the color wheel. For example, magenta is made with red and blue, and is the complement to green.
2. Tertiary colors are the result of combining a primary color with one of its secondary color neighbors. For example, red (primary) and yellow (secondary) make orange. This adds six more colors to a simple color wheel for a total of twelve.
3. There are various methods of selecting different colors that result in a harmonious combination in design. One example that can use tertiary colors is called the split-complementary color scheme. This scheme starts with a base color, then pairs it with the two colors that are adjacent to its complement. The three colors provide strong visual contrast in a design, but are more subtle than using two complementary colors.

Here are three colors created using the split-complement scheme:

| **Color** | **Hex Code** |
| --- | --- |
| orange | #FF7D00 |
| cyan | #00FFFF |
| raspberry | #FF007D |

1. 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.
2. **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.
3. **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.
4. **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%) |

1. The hsl()option in CSS also makes it easy to adjust the tone of a color. Mixing white with a pure hue creates a tint of that color, and adding black will make a shade. Alternatively, a tone is produced by adding gray or by both tinting and shading. Recall that the 's' and 'l' of hsl()stand for saturation and lightness, respectively. The saturation percent changes the amount of gray and the lightness percent determines how much white or black is in the color. This is useful when you have a base hue you like, but need different variations of it.
2. 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));

1. The repeating-linear-gradient()function is very similar to linear-gradient()with the major difference that it repeats the specified gradient pattern. repeating-linear-gradient()accepts a variety of values, but for simplicity, you'll work with an angle value and color stop values in this challenge.
2. The angle value is the direction of the gradient. Color stops are like width values that mark where a transition takes place, and are given with a percentage or a number of pixels.
3. 0px [yellow -- blend -- blue] 40px [green -- blend -- red] 80px
4. 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.

Using the url of https://i.imgur.com/MJAkxbh.png, set the background of the whole page with the body selector.

1. To change the scale of an element, CSS has the transformproperty, along with its scale()function. The following code example doubles the size of all the paragraph elements on the page:

p {  
  transform:scale(2);  
}

1. The next function of the transformproperty 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);  
}

1. Given that the skewX()function skews the selected element along the X-axis by a given degree, it is no surprise that the skewY()property skews an element along the Y (vertical) axis.
2. By manipulating different selectors and properties, you can make interesting shapes. One of the easier ones to try is a crescent moon shape. For this challenge you need to work with the box-shadow property that sets the shadow of an element, along with the border-radius property that controls the roundness of the element's corners.
3. You will create a round, transparent object with a crisp shadow that is slightly offset to the side - the shadow is actually going to be the moon shape you see.
4. In order to create a round object, the border-radius property should be set to a value of 50%.
5. You may recall from an earlier challenge that the box-shadow property takes values for offset-x, offset-y, blur-radius, spread-radius and a color value in that order. The blur-radius and spread-radius values are optional.
6. One of the most popular shapes in the world is the heart shape, and in this challenge you'll create one using pure CSS. But first, you need to understand the ::before and ::after pseudo-elements. These pseudo-elements are used to add something before or after a selected element. In the following example, a ::before pseudo-element is used to add a rectangle to an element with the class heart:
7. .heart::before {  
     content: "";  
     background-color: yellow;  
     border-radius: 25%;  
     position: absolute;  
     height: 50px;  
     width: 70px;  
     top: -50px;  
     left: 5px;  
   }
8. For the ::before and ::afterpseudo-elements to function properly, they must have a defined contentproperty. This property is usually used to add things like a photo or text to the selected element. When the ::before and ::after pseudo-elements are used to make shapes, the content property is still required, but it's set to an empty string.
9. 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:
10. animation-name sets the name of the animation, which is later used by @keyframes to tell CSS which rules go with which animations.
11. animation-duration sets the length of time for the animation.
12. @keyframesis 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 @keyframesand the animation properties:
13. #anim {  
      animation-name: colorful;  
      animation-duration: 3s;  
    }  
    @keyframes colorful {  
      0% {  
        background-color: blue;  
      }  
      100% {  
        background-color: yellow;  
      }  
    }
14. For the element with the animid, the code snippet above sets the animation-nameto colorfuland sets the animation-durationto 3 seconds. Then the @keyframesrule 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%.
15. You can use CSS @keyframesto change the color of a button in its hover state.
16. Here's an example of changing the width of an image on hover:
17. <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.

Notice how the animation resets after 500mshas 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-modeproperty to forwards. The animation-fill-modespecifies the style applied to an element when the animation has finished. You can set it like so:

animation-fill-mode: forwards;

1. When elements have a specified position, such as fixedor relative, the CSS offset properties right, left, top, and bottomcan be used in animation rules to create movement.
2. As shown in the example below, you can push the item downwards then upwards by setting the topproperty of the 50%keyframe to 50px, but having it set to 0px for the first (0%) and the last (100%) keyframe.
3. @keyframes rainbow {  
     0% {  
       background-color: blue;  
       top: 0px;  
     }  
     50% {  
       background-color: green;  
       top: 50px;  
     }  
     100% {  
       background-color: yellow;  
       top: 0px;  
     }  
   }

Create Visual Direction by Fading an Element from Left to Right

1. For this challenge, you'll change the opacity of an animated element so it gradually fades as it reaches the right side of the screen.
2. In the displayed animation, the round element with the gradient background moves to the right by the 50% mark of the animation per the @keyframes rule.

Animate Elements Continually Using an Infinite Animation Count

1. The previous challenges covered how to use some of the animation properties and the @keyframesrule. 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:
2. animation-iteration-count: 3;
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.

Make a CSS Heartbeat using an Infinite Animation Count

1. Here's one more continuous animation example with the animation-iteration-countproperty that uses the heart you designed in a previous challenge.
2. The one-second long heartbeat animation consists of two animated pieces. The heartelements (including the :beforeand :afterpieces) are animated to change size using the transformproperty, and the background divis animated to change its color using the backgroundproperty.

Animate Elements at Variable Rates

1. There are a variety of ways to alter the animation rates of similarly animated elements. So far, this has been achieved by applying an animation-iteration-count property and setting @keyframes rules.
2. To illustrate, the animation on the right consists of two "stars" that each decrease in size and opacity at the 20% mark in the @keyframes rule, which creates the twinkle animation. You can change the @keyframes rule for one of the elements so the stars twinkle at different rates.
3. Alter the animation rate for the element with the class name of star-1by changing its @keyframes rule to 50%.
4. Keep the heart beating by adding the animation-iteration-count property for both the back class and the heart class and setting the value to infinite. The heart:before and heart:after selectors do not need any animation properties.

Animate Multiple Elements at Variable Rates

In the previous challenge, you changed the animation rates for two similarly animated elements by altering their @keyframesrules. You can achieve the same goal by manipulating the animation-durationof multiple elements.

In the animation running in the code editor, there are three "stars" in the sky that twinkle at the same rate on a continuous loop. To make them twinkle at different rates, you can set the animation-durationproperty to different values for each element.