# Introduction to the Applied Visual Design Challenges

# Visual Design in web development is a broad topic. It combines typography, color theory, graphics, animation, and page layout to help deliver a site's message. The definition of good design is a well-discussed subject, with many books on the theme. At a basic level, most web content provides a user with information. The visual design of the page can influence its presentation and a user's experience. In web development, HTML gives structure and semantics to a page's content, and CSS controls the layout and appearance of it. This section covers some of the basic tools developers use to create their own visual designs.

1. [Create Visual Balance Using the text-align Property](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/create-visual-balance-using-the-text-align-property)

This section of the curriculum focuses on Applied Visual Design. The first group of challenges build on the given card layout to show a number of core principles.

Text is often a large part of web content. CSS has several options for how to align it with the text-align property.

text-align: justify; causes all lines of text except the last line to meet the left and right edges of the line box.

text-align: center; centers the text

text-align: right; right-aligns the text

And text-align: left; (the default) left-aligns the text.

Align the h4 tag's text, which says "Google", to the center. Then justify the paragraph tag which contains information about how Google was founded.

<style>

  h4 {

text-align: center;

  }

  p {

text-align: justify;

  }

  .links {

    margin-right: 20px;

  }

  .fullCard {

    border: 1px solid #ccc;

    border-radius: 5px;

    margin: 10px 5px;

    padding: 4px;

  }

  .cardContent {

    padding: 10px;

  }

</style>

1. [Adjust the Width of an Element Using the width Property](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/adjust-the-width-of-an-element-using-the-width-property)

You can specify the width of an element using the width property in CSS. Values can be given in relative length units (such as em), absolute length units (such as px), or as a percentage of its containing parent element. Here's an example that changes the width of an image to 220px:

img {

width: 220px;

}

Add a width property to the entire card and set it to an absolute value of 245px. Use the fullCard class to select the element.

<style>

  h4 {

    text-align: center;

  }

  p {

    text-align: justify;

  }

  .links {

    margin-right: 20px;

    text-align: left;

  }

  .fullCard {

    width: 245px;

    border: 1px solid #ccc;

    border-radius: 5px;

    margin: 10px 5px;

    padding: 4px;

  }

1. [Adjust the Height of an Element Using the height Property](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/adjust-the-height-of-an-element-using-the-height-property)

You can specify the height of an element using the height property in CSS, similar to the width property. Here's an example that changes the height of an image to 20px:

img {

height: 20px;

}

Add a height property to the h4 tag and set it to 25px.

**Note:** You may need to be at 100% zoom to pass the test on this challenge.

<style>

  h4 {

    text-align: center;

  height: 25px;

  }

1. [Use the strong Tag to Make Text Bold](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-the-strong-tag-to-make-text-bold)

To make text bold, you can use the strong tag. This is often used to draw attention to text and symbolize that it is important. With the strong tag, the browser applies the CSS of font-weight: bold; to the element.

Wrap a strong tag around "Stanford University" inside the p tag (do not include the period).

<p>Google was founded by Larry Page and Sergey Brin while they were Ph.D. students at <strong>Stanford University</strong>.</p>

1. [Use the u Tag to Underline Text](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-the-u-tag-to-underline-text)

To underline text, you can use the u tag. This is often used to signify that a section of text is important, or something to remember. With the u tag, the browser applies the CSS of text-decoration: underline; to the element.

Wrap the u tag only around the text "Ph.D. students".

**Note:** Try to avoid using the u tag when it could be confused for a link. Anchor tags also have a default underlined formatting.

<p>Google was founded by Larry Page and Sergey Brin while they were <u>Ph.D. students</u> at <strong>Stanford University</strong>.</p>

1. [Use the em Tag to Italicize Text](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-the-em-tag-to-italicize-text)

To emphasize text, you can use the em tag. This displays text as italicized, as the browser applies the CSS of font-style: italic; to the element.

Wrap an em tag around the contents of the paragraph tag to give it emphasis.

<p><em>Google was founded by Larry Page and Sergey Brin while they were <u>Ph.D. students</u> at <strong>Stanford University</strong>.</em></p>

1. [Use the s Tag to Strikethrough Text](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-the-s-tag-to-strikethrough-text)

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.

Wrap the s tag around "Google" inside the h4 tag and then add the word "Alphabet" beside it, which should not have the strikethrough formatting.

<h4><s>Google</s>Alphabet</h4>

1. [Create a Horizontal Line Using the hr Element](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/create-a-horizontal-line-using-the-hr-element)

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.

Add an hr tag underneath the h4 which contains the card title.

**Note:** In HTML, hr is a self-closing tag, and therefore doesn't need a separate closing tag.

<h4><s>Google</s>Alphabet</h4>

<hr>

      <p><em>Google was founded by Larry Page and Sergey Brin while they were <u>Ph.D. students</u> at <strong>Stanford University</strong>.</em></p>

1. [Adjust the background-color Property of Text](https://www.freecodecamp.org/learn/responsive-web-design/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.

To make the text stand out more, adjust the background-color of the h4 element to the given rgba() value.

Also for the h4, remove the height property and add padding of 10px.

<style>

  h4 {

    text-align: center;

    padding: 10px;

   background-color: rgba(45, 45, 45, 0.1)

  }

1. [Adjust the Size of a Header Versus a Paragraph Tag](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/adjust-the-size-of-a-header-versus-a-paragraph-tag)

The font size of header tags (h1 through h6) should generally be larger than the font size of paragraph tags. This makes it easier for the user to visually understand the layout and level of importance of everything on the page. You use the font-size property to adjust the size of the text in an element.

To make the heading significantly larger than the paragraph, change the font-size of the h4 tag to 27 pixels.

<style>

  h4 {

    text-align: center;

    background-color: rgba(45, 45, 45, 0.1);

    padding: 10px;

    font-size: 27px;

  }

1. [Add a box-shadow to a Card-like Element](https://www.freecodecamp.org/learn/responsive-web-design/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);

The element now has an id of thumbnail. With this selector, use the example CSS values above to place a box-shadow on the card.

<style>

#thumbnail {

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

}

</style>

<div class="fullCard" id="thumbnail">

1. [Decrease the Opacity of an Element](https://www.freecodecamp.org/learn/responsive-web-design/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.

Set the opacity of the anchor tags to 0.7 using links class to select them.

<style>

  .links {

    text-align: left;

    color: black;

    opacity: 0.7;

    <div class="cardLinks">

      <a href="https://en.wikipedia.org/wiki/Larry\_Page" target="\_blank" class="links">Larry Page</a><br><br>

      <a href="https://en.wikipedia.org/wiki/Sergey\_Brin" target="\_blank" class="links">Sergey Brin</a>

    </div>

1. [Use the text-transform Property to Make Text Uppercase](https://www.freecodecamp.org/learn/responsive-web-design/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 |

Transform the text of the h4 to be uppercase using the text-transform property.

<style>

  h4 {

    text-align: center;

    background-color: rgba(45, 45, 45, 0.1);

    padding: 10px;

    font-size: 27px;

    text-transform: uppercase;

  }

1. [Set the font-size for Multiple Heading Elements](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/set-the-font-size-for-multiple-heading-elements)

The font-size property is used to specify how large the text is in a given element. This rule can be used for multiple elements to create visual consistency of text on a page. In this challenge, you'll set the values for all h1 through h6 tags to balance the heading sizes.

Set the font-size of the h1 tag to 68px.

. . .

Set the font-size of the h6 tag to 14px.

<style>

h1 {

    font-size: 68px

}

. . .

h6 {

    font-size: 14px

}

</style>

<h1>This is h1 text</h1>

. . .

<h6>This is h6 text</h6>

1. [Set the font-weight for Multiple Heading Elements](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/set-the-font-weight-for-multiple-heading-elements)

You set the font-size of each heading tag in the last challenge, here you'll adjust the font-weight.

The font-weight property sets how thick or thin characters are in a section of text.

Set the font-weight of the h1 tag to 800.

. . .

Set the font-weight of the h6 tag to 200.

<style>

  h1 {

    font-size: 68px;

    font-weight: 800;

  }

  . . .

  h6 {

    font-size: 14px;

    font-weight: 200;

  }

</style>

<h1>This is h1 text</h1>

. . .

<h6>This is h6 text</h6>

1. [Set the font-size of Paragraph Text](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/set-the-font-size-of-paragraph-text)

The font-size property in CSS is not limited to headings, it can be applied to any element containing text.

Change the value of the font-size property for the paragraph to 16px to make it more visible.

<style>

  p {

    font-size: 16px;

  }

</style>

<p>

  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tem

. . .

re eu fugiat nulla pariatur.

</p>

1. [Set the line-height of Paragraphs](https://www.freecodecamp.org/learn/responsive-web-design/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.

Add a line-height property to the p tag and set it to 25px.

<style>

  p {

    font-size: 16px;

    line-height: 25px;

  }

</style>

<p>

  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tem

. . .

re eu fugiat nulla pariatur.

</p>

1. [Adjust the Hover State of an Anchor Tag](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/adjust-the-hover-state-of-an-anchor-tag)

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;

}

The code editor has a CSS rule to style all a tags black. Add a rule so that when the user hovers over the a tag, the color is blue.

<style>

  a {

    color: #000;

  }

a:hover {

  color: blue;

}

</style>

<a href="https://freecatphotoapp.com/" target="\_blank">CatPhotoApp</a>

1. [Change an Element's Relative Position](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/change-an-elements-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.

Change the position of the h2 to relative, and use a CSS offset to move it 15 pixels away from the top of where it sits in the normal flow. Notice there is no impact on the positions of the surrounding h1 and p elements.

<style>

  h2 {

position: relative;

   top: 15px;

  }

</style>

<body>

  <h1>On Being Well-Positioned</h1>

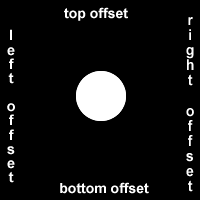
  <h2>Move me!</h2>

  <p>I still think the h2 is where it normally sits.</p>

</body>

1. [Move a Relatively Positioned Element with CSS Offsets](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/move-a-relatively-positioned-element-with-css-offsets)

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.



Use CSS offsets to move the h2 15 pixels to the right and 10 pixels up.

<head>

<style>

  h2 {

    position: relative;

    left: 15px;

    bottom: 10px;

  }

</style>

</head>

<body>

  <h1>On Being Well-Positioned</h1>

  <h2>Move me!</h2>

  <p>I still think the h2 is where it normally sits.</p>

</body>

1. [Lock an Element to its Parent with Absolute Positioning](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/lock-an-element-to-its-parent-with-absolute-positioning)

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.

Lock the #searchbar element to the top-right of its section parent by declaring its position as absolute. Give it top and right offsets of 50 pixels each.

<style>

  #searchbar {

position: absolute;

top: 50px;

right: 50px;

  }

  section {

    position: relative;

  }

</style>

<body>

  <h1>Welcome!</h1>

  <section>

    <form id="searchbar">

      <label for="search">Search:</label>

      <input type="search" id="search" name="search">

      <input type="submit" name="submit" value="Go!">

    </form>

  </section>

</body>

1. [Lock an Element to the Browser Window with Fixed Positioning](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/lock-an-element-to-the-browser-window-with-fixed-positioning)

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.

The navigation bar in the code is labeled with an id of navbar. Change its position to fixed, and offset it 0 pixels from the top and 0 pixels from the left. After you have added the code, scroll the preview window to see how the navigation stays in place.

<style>

  body {

    min-height: 150vh;

  }

  #navbar {

    position: fixed;

    top: 0px;

    left: 0px;

    width: 100%;

    background-color: #767676;

  }

  nav ul {

    margin: 0px;

    padding: 5px 0px 5px 30px;

  }

  nav li {

    display: inline;

    margin-right: 20px;

  }

  a {

    text-decoration: none;

  }

</style>

<body>

  <header>

    <h1>Welcome!</h1>

    <nav id="navbar">

      <ul>

        <li><a href="">Home</a></li>

        <li><a href="">Contact</a></li>

      </ul>

    </nav>

  </header>

  <p>I shift up when the #navbar is fixed to the browser window.</p>

</body>

1. [Push Elements Left or Right with the float Property](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/push-elements-left-or-right-with-the-float-property)

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.

The given markup would work well as a two-column layout, with the section and aside elements next to each other. Give the #left item a float of left and the #right item a float of right.

<head>

  <style>

    #left {

      float: left;

      width: 50%;

    }

    #right {

      float: right;

      width: 40%;

    }

    aside, section {

      padding: 2px;

      background-color: #ccc;

    }

  </style>

</head>

1. [Change the Position of Overlapping Elements with the z-index Property](https://www.freecodecamp.org/learn/responsive-web-design/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).

<style>

  div {

    width: 60%;

    height: 200px;

    margin-top: 20px;

  }

  .first {

    background-color: red;

    position: absolute;

    z-index: 2;

  }

  .second {

    background-color: blue;

    position: absolute;

    left: 40px;

    top: 50px;

    z-index: 1;

  }

</style>

<div class="first"></div>

<div class="second"></div>

1. [Center an Element Horizontally Using the margin Property](https://www.freecodecamp.org/learn/responsive-web-design/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.

Center the div on the page by adding a margin property with a value of auto.

<style>

  div {

    background-color: blue;

    height: 100px;

    width: 100px;

    margin: auto;

  }

</style>

<div></div>

1. [Learn about Complementary Colors](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/learn-about-complementary-colors)

Color theory and its impact on design is a deep topic and only the basics are covered in the following challenges. On a website, color can draw attention to content, evoke emotions, or create visual harmony. Using different combinations of colors can really change the look of a website, and a lot of thought can go into picking a color palette that works with your content.

The color wheel is a useful tool to visualize how colors relate to each other - it's a circle where similar hues are neighbors and different hues are farther apart. 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)

This is different than the outdated RYB color model that many of us were taught in school, which has different primary and complementary colors. Modern color theory uses the additive RGB model (like on a computer screen) and the subtractive CMY(K) model (like in printing). Read [here](https://en.wikipedia.org/wiki/Color_model) for more information on this complex subject.

There are many color picking tools available online that have an option to find the complement of a color.

**Note:** For all color challenges: Using color can be a powerful way to add visual interest to a page. However, color alone should not be used as the only way to convey important information because users with visual impairments may not understand that content. This issue will be covered in more detail in the Applied Accessibility challenges.

Change the background-color property of the blue and yellow classes to their respective colors. Notice how the colors look different next to each other than they do compared against the white background.

<style>

  body {

    background-color: #FFFFFF;

  }

  .blue {

    background-color: #0000ff;

  }

  .yellow {

    background-color: #ffff00;

  }

  div {

    display: inline-block;

    height: 100px;

    width: 100px;

  }

</style>

<div class="blue"></div>

<div class="yellow"></div>

1. [Learn about Tertiary Colors](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/learn-about-tertiary-colors)

Computer monitors and device screens create different colors by combining amounts of red, green, and blue light. This is known as the RGB additive color model in modern color theory. 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). You saw these colors in the Complementary Colors challenge. 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.

Tertiary colors are the result of combining a primary color with one of its secondary color neighbors. For example, within the RGB color model, red (primary) and yellow (secondary) make orange (tertiary). This adds six more colors to a simple color wheel for a total of twelve.

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 | #FF7F00 |
| cyan | #00FFFF |
| raspberry | #FF007F |

Change the background-color property of the orange, cyan, and raspberry classes to their respective colors. Make sure to use the hex codes and not the color names.

<style>

  body {

    background-color: #FFFFFF;

  }

  .orange {

    background-color: #ff7f00;

  }

  .cyan {

    background-color: #00ffff;

  }

  .raspberry {

    background-color: #ff007f;

  }

  div {

    height: 100px;

    width: 100px;

    margin-bottom: 5px;

  }

</style>

<div class="orange"></div>

<div class="cyan"></div>

<div class="raspberry"></div>

1. [Adjust the Color of Various Elements to Complementary Colors](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/adjust-the-color-of-various-elements-to-complementary-colors)

The Complementary Colors challenge showed that opposite colors on the color wheel can make each other appear more vibrant when placed side-by-side. However, the strong visual contrast can be jarring if it's overused on a website, and can sometimes make text harder to read if it's placed on a complementary-colored background. In practice, one of the colors is usually dominant and the complement is used to bring visual attention to certain content on the page.

This page will use a shade of teal (#09A7A1) as the dominant color, and its orange (#FF790E) complement to visually highlight the sign-up buttons. Change the background-color of both the header and footer from black to the teal color. Then change the h2 text color to teal as well. Finally, change the background-color of the button to the orange color.

<style>

  body {

    background-color: white;

  }

  header {

    background-color: #09A7A1;

    color: white;

    padding: 0.25em;

  }

  h2 {

    color: #09A7A1;

  }

  button {

    background-color: #FF790E;

  }

  footer {

    background-color: #09A7A1;

    color: white;

    padding: 0.5em;

  }

</style>

<header>

  <h1>Cooking with FCC!</h1>

</header>

<main>

  <article>

    <h2>Machine Learning in the Kitchen</h2>

    <p>Join this two day workshop that walks through how to implement cutting-edge snack-getting algorithms with a command line interface. Coding usually involves writing exact instructions, but sometimes you need your computer to execute flexible commands, like <code>fetch Pringles</code>.</p>

    <button>Sign Up</button>

  </article>

  <article>

    <h2>Bisection Vegetable Chopping</h2>

    <p>This week-long retreat will level-up your coding ninja skills to actual ninja skills. No longer is the humble bisection search limited to sorted arrays or coding interview questions, applying its concepts in the kitchen will have you chopping carrots in O(log n) time before you know it.</p>

    <button>Sign Up</button>

  </article>

</main>

<br>

<footer>&copy; 2018 FCC Kitchen</footer>

1. [Adjust the Hue of a Color](https://www.freecodecamp.org/learn/responsive-web-design/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.

<style>

  body {

    background-color: #FFFFFF;

  }

  .green {

    background-color: hsl(120, 100%, 50%);

  }

  .cyan {

    background-color: hsl(180, 100%, 50%);

  }

  .blue {

    background-color: hsl(240, 100%, 50%);

  }

1. [Adjust the Tone of a Color](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/adjust-the-tone-of-a-color)

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.

All elements have a default background-color of transparent. Our nav element currently appears to have a cyan background, because the element behind it has a background-color set to cyan. Add a background-color to the nav element so it uses the same cyan hue, but has 80% saturation and 25% lightness values to change its tone and shade.

<style>

  header {

    background-color: hsl(180, 90%, 35%);

    color: #FFFFFF;

  }

  nav {

    background-color: hsl(180, 80%, 25%);

  }

1. [Create a Gradual CSS Linear Gradient](https://www.freecodecamp.org/learn/responsive-web-design/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 horizontal gradient (from left to right) 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));

Use a linear-gradient() for the div element's background, and set it from a direction of 35 degrees to change the color from #CCFFFF to #FFCCCC.

<style>

  div {

    border-radius: 20px;

    width: 70%;

    height: 400px;

    margin: 50px auto;

    background: linear-gradient(35deg, #CCFFFF, #FFCCCC);

  }

</style>

<div></div>

1. [Use a CSS Linear Gradient to Create a Striped Element](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-a-css-linear-gradient-to-create-a-striped-element)

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.

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.

In the example demonstrated in the code editor, the gradient starts with the color yellow at 0 pixels which blends into the second color blue at 40 pixels away from the start. Since the next color stop is also at 40 pixels, the gradient immediately changes to the third color green, which itself blends into the fourth color value red as that is 80 pixels away from the beginning of the gradient.

For this example, it helps to think about the color stops as pairs where every two colors blend together.

0px [yellow -- blend -- blue] 40px [green -- blend -- red] 80px

If every two color stop values are the same color, the blending isn't noticeable because it's between the same color, followed by a hard transition to the next color, so you end up with stripes.

Make stripes by changing the repeating-linear-gradient() to use a gradient angle of 45deg, then set the first two color stops to yellow, and finally the second two color stops to black.

<style>

  div{

    border-radius: 20px;

    width: 70%;

    height: 400px;

    margin:  50 auto;

    background: repeating-linear-gradient(

      45deg,

      yellow 0px,

      yellow 40px,

      black 40px,

      black 80px

    );

  }

</style>

<div></div>

1. [Create Texture by Adding a Subtle Pattern as a Background Image](https://www.freecodecamp.org/learn/responsive-web-design/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.

Using the url of https://cdn-media-1.freecodecamp.org/imgr/MJAkxbh.png, set the background of the whole page with the body selector.

<style>

  body {

    backbackground: url(https://cdn-media-1.freecodecamp.org/imgr/MJAkxbh.png);

  }

</style>

1. [Use the CSS Transform scale Property to Change the Size of an Element](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-the-css-transform-scale-property-to-change-the-size-of-an-element)

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

p {

transform: scale(2);

}

Increase the size of the element with the id of ball2 to 1.5 times its original size.

<style>

  .ball {

    width: 40px;

    height: 40px;

    margin: 50 auto;

    position: fixed;

    background: linear-gradient(

      35deg,

      #ccffff,

      #ffcccc

    );

    border-radius: 50%;

  }

  #ball1 {

    left: 20%;

  }

  #ball2 {

    left: 65%;

    transform: scale(1.5);

  }

</style>

<div class="ball" id= "ball1"></div>

<div class="ball" id= "ball2"></div>

1. [Use the CSS Transform scale Property to Scale an Element on Hover](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-the-css-transform-scale-property-to-scale-an-element-on-hover)

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.

Add a CSS rule for the hover state of the div and use the transform property to scale the div element to 1.1 times its original size when a user hovers over it.

<style>

  div {

    width: 70%;

    height: 100px;

    margin:  50px auto;

    background: linear-gradient(

      53deg,

      #ccfffc,

      #ffcccf

    );

  }

  div:hover {

    transform: scale(1.1);

  }

</style>

<div></div>

1. [Use the CSS Transform Property skewX to Skew an Element Along the X-Axis](https://www.freecodecamp.org/learn/responsive-web-design/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);

}

Skew the element with the id of bottom by 24 degrees along the X-axis by using the transform property.

<style>

  div {

    width: 70%;

    height: 100px;

    margin:  50px auto;

  }

  #top {

    background-color: red;

  }

  #bottom {

    background-color: blue;

    transform: skewX(24deg);

  }

</style>

<div id="top"></div>

<div id="bottom"></div>

1. [Use the CSS Transform Property skewY to Skew an Element Along the Y-Axis](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-the-css-transform-property-skewy-to-skew-an-element-along-the-y-axis)

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.

Skew the element with the id of top -10 degrees along the Y-axis by using the transform property.

<style>

  div {

    width: 70%;

    height: 100px;

    margin: 50px auto;

  }

  #top {

    background-color: red;

    transform: skewY(-10deg);

  }

  #bottom {

    background-color: blue;

    transform: skewX(24deg);

  }

</style>

<div id="top"></div>

<div id="bottom"></div>

1. [Create a Graphic Using CSS](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/create-a-graphic-using-css)

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.

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.

In order to create a round object, the border-radius property should be set to a value of 50%.

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.

Manipulate the square element in the editor to create the moon shape. First, change the background-color to transparent, then set the border-radius property to 50% to make the circular shape. Finally, change the box-shadow property to set the offset-x to 25px, the offset-y to 10px, blur-radius to 0, spread-radius to 0, and color to blue.

<style>

  .center {

    position: absolute;

    margin: auto;

    top: 0;

    right: 0;

    bottom: 0;

    left: 0;

    width: 100px;

    height: 100px;

    background-color: transparent;

    border-radius: 50%;

    box-shadow: 25px 10px 0px 0px blue;

  }

</style>

<div class="center"></div>

1. [Create a More Complex Shape Using CSS and HTML](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/create-a-more-complex-shape-using-css-and-html)

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:

.heart::before {

content: "";

background-color: yellow;

border-radius: 25%;

position: absolute;

height: 50px;

width: 70px;

top: -50px;

left: 5px;

}

For the ::before and ::after pseudo-elements to function properly, they must have a defined content property. 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. In the above example, the element with the class of heart has a ::before pseudo-element that produces a yellow rectangle with height and width of 50px and 70px, respectively. This rectangle has round corners due to its 25% border radius and is positioned absolutely at 5px from the left and 50px above the top of the element.

Transform the element on the screen to a heart. In the heart::after selector, change the background-color to pink and the border-radius to 50%.

Next, target the element with the class heart (just heart) and fill in the transform property. Use the rotate() function with -45 degrees.

Finally, in the heart::before selector, set its content property to an empty string.

<style>

  .heart {

    position: absolute;

    margin: auto;

    top: 0;

    right: 0;

    bottom: 0;

    left: 0;

    background-color: pink;

    height: 50px;

    width: 50px;

    transform: rotate(-45deg);

  }

  .heart::after {

    background-color: pink;

    content: "";

    border-radius: 50%;

    position: absolute;

    width: 50px;

    height: 50px;

    top: 0px;

    left: 25px;

  }

  .heart::before {

    content: "";

    background-color: pink;

    border-radius: 50%;

    position: absolute;

    width: 50px;

    height: 50px;

    top: -25px;

    left: 0px;

  }

</style>

<div class="heart"></div>

1. [Learn How the CSS @keyframes and animation Properties Work](https://www.freecodecamp.org/learn/responsive-web-design/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%.

Create an animation for the element with the id rect, by setting the animation-name to rainbow and the animation-duration to 4 seconds. Next, declare a @keyframes rule, and set the background-color at the beginning of the animation (0%) to blue, the middle of the animation (50%) to green, and the end of the animation (100%) to yellow.

<style>

  div {

    height: 40px;

    width: 70%;

    background: black;

    margin: 50px auto;

    border-radius: 5px;

  }

  #rect {

    animation-name: rainbow;

    animation-duration: 4s;

  }

  @keyframes rainbow {

    0% {

      background-color: blue;

    }

    50% {

      background-color: green;

    }

    100% {

      background-color: yellow;

    }

  }

</style>

<div id="rect"></div>

1. [Use CSS Animation to Change the Hover State of a Button](https://www.freecodecamp.org/learn/responsive-web-design/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%.

<style>

  button {

    border-radius: 5px;

    color: white;

    background-color: #0F5897;

    padding: 5px 10px 8px 10px;

  }

  button:hover {

    animation-name: background-color;

    animation-duration: 500ms;

  }

 @keyframes background-color {

    100% {

      background-color: #4791d0;

    }

  }

</style>

<button>Register</button>

1. [Modify Fill Mode of an Animation](https://www.freecodecamp.org/learn/responsive-web-design/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;

Set the animation-fill-mode property of button:hover to forwards so the button stays highlighted when a user hovers over it.

<style>

  button {

    border-radius: 5px;

    color: white;

    background-color: #0F5897;

    padding: 5px 10px 8px 10px;

  }

  button:hover {

    animation-name: background-color;

    animation-duration: 500ms;

    /\* Only change code below this line \*/

    animation-fill-mode: forwards;

    /\* Only change code above this line \*/

  }

  @keyframes background-color {

    100% {

      background-color: #4791d0;

    }

  }

</style>

<button>Register</button>

1. [Create Movement Using CSS Animation](https://www.freecodecamp.org/learn/responsive-web-design/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.

<style>

  div {

    height: 40px;

    width: 70%;

    background: black;

    margin: 50px auto;

    border-radius: 5px;

    position: relative;

  }

  #rect {

    animation-name: rainbow;

    animation-duration: 4s;

  }

  @keyframes rainbow {

    0% {

      background-color: blue;

      left: 0px;

    }

    50% {

      background-color: green;

      left: 25px;

    }

    100% {

      background-color: yellow;

      left: -25px;

    }

  }

</style>

<div id="rect"></div>

1. [Create Visual Direction by Fading an Element from Left to Right](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/create-visual-direction-by-fading-an-element-from-left-to-right)

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.

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.

Target the element with the id of ball and add the opacity property set to 0.1 at 50%, so the element fades as it moves to the right.

<style>

  #ball {

    width: 70px;

    height: 70px;

    margin: 50px auto;

    position: fixed;

    left: 20%;

    border-radius: 50%;

    background: linear-gradient(

      35deg,

      #ccffff,

      #ffcccc

    );

    animation-name: fade;

    animation-duration: 3s;

  }

  @keyframes fade {

    50% {

      left: 60%;

      opacity: 0.1;

    }

  }

</style>

<div id="ball"></div>

1. [Animate Elements Continually Using an Infinite Animation Count](https://www.freecodecamp.org/learn/responsive-web-design/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.

To keep the ball bouncing on the right on a continuous loop, change the animation-iteration-count property to infinite.

<style>

  #ball {

    width: 100px;

    height: 100px;

    margin: 50px auto;

    position: relative;

    border-radius: 50%;

    background: linear-gradient(

      35deg,

      #ccffff,

      #ffcccc

    );

    animation-name: bounce;

    animation-duration: 1s;

    animation-iteration-count: infinite;

  }

  @keyframes bounce{

    0% {

      top: 0px;

    }

    50% {

      top: 249px;

      width: 130px;

      height: 70px;

    }

    100% {

      top: 0px;

    }

  }

</style>

<div id="ball"></div>

1. [Make a CSS Heartbeat using an Infinite Animation Count](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/make-a-css-heartbeat-using-an-infinite-animation-count)

Here's one more continuous animation example with the animation-iteration-count property that uses the heart you designed in a previous challenge.

The one-second long heartbeat animation consists of two animated pieces. The heart elements (including the :before and :after pieces) are animated to change size using the transform property, and the background div is animated to change its color using the background property.

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.

<style>

  .back {

    position: fixed;

    padding: 0;

    margin: 0;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: white;

    animation-name: backdiv;

    animation-duration: 1s;

    animation-iteration-count: infinite;

  }

  .heart {

    position: absolute;

    margin: auto;

    top: 0;

    right: 0;

    bottom: 0;

    left: 0;

    background-color: pink;

    height: 50px;

    width: 50px;

    transform: rotate(-45deg);

    animation-name: beat;

    animation-duration: 1s;

    animation-iteration-count: infinite;

  }

  .heart:after {

    background-color: pink;

    content: "";

    border-radius: 50%;

    position: absolute;

    width: 50px;

    height: 50px;

    top: 0px;

    left: 25px;

    }

  .heart:before {

    background-color: pink;

    content: "";

    border-radius: 50%;

    position: absolute;

    width: 50px;

    height: 50px;

    top: -25px;

    left: 0px;

    }

  @keyframes backdiv {

    50% {

      background: #ffe6f2;

    }

  }

  @keyframes beat {

    0% {

      transform: scale(1) rotate(-45deg);

    }

    50% {

      transform: scale(0.6) rotate(-45deg);

    }

  }

</style>

<div class="back"></div>

<div class="heart"></div

1. [Animate Elements at Variable Rates](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/animate-elements-at-variable-rates)

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.

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.

Alter the animation rate for the element with the class name of star-1 by changing its @keyframes rule to 50%.

<style>

  .stars {

    background-color: white;

    height: 30px;

    width: 30px;

    border-radius: 50%;

    animation-iteration-count: infinite;

  }

  .star-1 {

    margin-top: 15%;

    margin-left: 60%;

    animation-name: twinkle-1;

    animation-duration: 1s;

  }

  .star-2 {

    margin-top: 25%;

    margin-left: 25%;

    animation-name: twinkle-2;

    animation-duration: 1s;

  }

  @keyframes twinkle-1 {

    50% {

      transform: scale(0.5);

      opacity: 0.5;

    }

  }

  @keyframes twinkle-2 {

    50% {

      transform: scale(0.5);

      opacity: 0.5;

    }

  }

  #back {

    position: fixed;

    padding: 0;

    margin: 0;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: linear-gradient(black, #000099, #66c2ff, #ffcccc, #ffeee6);

  }

</style>

<div id="back"></div>

<div class="star-1 stars"></div>

<div class="star-2 stars"></div>

1. [Animate Multiple Elements at Variable Rates](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/animate-multiple-elements-at-variable-rates)

In the previous challenge, you changed the animation rates for two similarly animated elements by altering their @keyframes rules. You can achieve the same goal by manipulating the animation-duration of 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-duration property to different values for each element.

Set the animation-duration of the elements with the classes star-1, star-2, and star-3 to 1s, 0.9s, and 1.1s, respectively.

<style>

  .stars {

    background-color: white;

    height: 30px;

    width: 30px;

    border-radius: 50%;

    animation-iteration-count: infinite;

  }

  .star-1 {

    margin-top: 15%;

    margin-left: 60%;

    animation-duration: 1s;

    animation-name: twinkle;

  }

  .star-2 {

    margin-top: 25%;

    margin-left: 25%;

    animation-duration: 0.9s;

    animation-name: twinkle;

  }

  .star-3 {

    margin-top: 10%;

    margin-left: 50%;

    animation-duration: 1.1s;

    animation-name: twinkle;

  }

  @keyframes twinkle {

    20% {

      transform: scale(0.5);

      opacity: 0.5;

    }

  }

  #back {

    position: fixed;

    padding: 0;

    margin: 0;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: linear-gradient(black, #000099, #66c2ff, #ffcccc, #ffeee6);

  }

</style>

<div id="back"></div>

<div class="star-1 stars"></div>

<div class="star-2 stars"></div>

<div class="star-3 stars"></div>

1. [Change Animation Timing with Keywords](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/change-animation-timing-with-keywords)

n 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.

<style>

  .balls {

    border-radius: 50%;

    background: linear-gradient(

      35deg,

      #ccffff,

      #ffcccc

    );

    position: fixed;

    width: 50px;

    height: 50px;

    margin-top: 50px;

    animation-name: bounce;

    animation-duration: 2s;

    animation-iteration-count: infinite;

  }

  #ball1 {

    left:27%;

    animation-timing-function: linear;

  }

  #ball2 {

    left:56%;

    animation-timing-function: ease-out;

  }

  @keyframes bounce {

    0% {

      top: 0px;

    }

    100% {

      top: 249px;

    }

  }

</style>

<div class="balls" id="ball1"></div>

<div class="balls" id="ball2"></div>

1. [Learn How Bezier Curves Work](https://www.freecodecamp.org/learn/responsive-web-design/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.

For the element with the id of ball1, change the value of the animation-timing-function property from linear to its equivalent cubic-bezier function value. Use the point values given in the example above.

<style>

  .balls{

    border-radius: 50%;

    background: linear-gradient(

      35deg,

      #ccffff,

      #ffcccc

    );

    position: fixed;

    width: 50px;

    height: 50px;

    margin-top: 50px;

    animation-name: bounce;

    animation-duration: 2s;

    animation-iteration-count: infinite;

  }

  #ball1 {

    left: 27%;

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

  }

  #ball2 {

    left: 56%;

    animation-timing-function: ease-out;

  }

  @keyframes bounce {

    0% {

      top: 0px;

    }

    100% {

      top: 249px;

    }

  }

</style>

<div class="balls" id="ball1"></div>

<div class="balls" id="ball2"></div>

1. [Use a Bezier Curve to Move a Graphic](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/use-a-bezier-curve-to-move-a-graphic)

A previous challenge discussed the ease-out keyword that describes an animation change that speeds up first and then slows down at the end of the animation. On the right, the difference between the ease-out keyword (for the blue element) and linear keyword (for the red element) is demonstrated. Similar animation progressions to the ease-out keyword can be achieved by using a custom cubic Bezier curve function.

In general, changing the p1 and p2 anchor points drives the creation of different Bezier curves, which controls how the animation progresses through time. Here's an example of a Bezier curve using values to mimic the ease-out style:

animation-timing-function: cubic-bezier(0, 0, 0.58, 1);

Remember that all cubic-bezier functions start with p0 at (0, 0) and end with p3 at (1, 1). In this example, the curve moves faster through the Y-axis (starts at 0, goes to p1 y value of 0, then goes to p2 y value of 1) than it moves through the X-axis (0 to start, then 0 for p1, up to 0.58 for p2). As a result, the change in the animated element progresses faster than the time of the animation for that segment. Towards the end of the curve, the relationship between the change in x and y values reverses - the y value moves from 1 to 1 (no change), and the x values move from 0.58 to 1, making the animation changes progress slower compared to the animation duration.

To see the effect of this Bezier curve in action, change the animation-timing-function of the element with id of red to a cubic-bezier function with x1, y1, x2, y2 values set respectively to 0, 0, 0.58, 1. This will make both elements progress through the animation similarly.

<style>

  .balls{

    border-radius: 50%;

    position: fixed;

    width: 50px;

    height: 50px;

    margin-top: 50px;

    animation-name: bounce;

    animation-duration: 2s;

    animation-iteration-count: infinite;

  }

  #red {

    background: red;

    left: 27%;

    animation-timing-function: cubic-bezier(0, 0, 0.58, 1);

  }

  #blue {

    background: blue;

    left: 56%;

    animation-timing-function: ease-out;

  }

  @keyframes bounce {

    0% {

      top: 0px;

    }

    100% {

      top: 249px;

    }

  }

</style>

<div class="balls" id= "red"></div>

<div class="balls" id= "blue"></div>

1. [Make Motion More Natural Using a Bezier Curve](https://www.freecodecamp.org/learn/responsive-web-design/applied-visual-design/make-motion-more-natural-using-a-bezier-curve)

This challenge animates an element to replicate the movement of a ball being juggled. Prior challenges covered the linear and ease-out cubic Bezier curves, however neither depicts the juggling movement accurately. You need to customize a Bezier curve for this.

The animation-timing-function automatically loops at every keyframe when the animation-iteration-count is set to infinite. Since there is a keyframe rule set in the middle of the animation duration (at 50%), it results in two identical animation progressions at the upward and downward movement of the ball.

The following cubic Bezier curve simulates a juggling movement:

cubic-bezier(0.3, 0.4, 0.5, 1.6);

Notice that the value of y2 is larger than 1. Although the cubic Bezier curve is mapped on a 1 by 1 coordinate system, and it can only accept x values from 0 to 1, the y value can be set to numbers larger than one. This results in a bouncing movement that is ideal for simulating the juggling ball.

Change value of the animation-timing-function of the element with the id of green to a cubic-bezier function with x1, y1, x2, y2 values set respectively to 0.311, 0.441, 0.444, 1.649.

<style>

  .balls {

    border-radius: 50%;

    position: fixed;

    width: 50px;

    height: 50px;

    top: 60%;

    animation-name: jump;

    animation-duration: 2s;

    animation-iteration-count: infinite;

  }

  #red {

    background: red;

    left: 25%;

    animation-timing-function: linear;

  }

  #blue {

    background: blue;

    left: 50%;

    animation-timing-function: ease-out;

  }

  #green {

    background: green;

    left: 75%;

    animation-timing-function: cubic-bezier(0.311, 0.441, 0.444, 1.649);

  }

  @keyframes jump {

    50% {

      top: 10%;

    }

  }

</style>

<div class="balls" id="red"></div>

<div class="balls" id="blue"></div>

<div class="balls" id="green"></div>