**CCS www.freecodecamp.org**

**DISEÑO VISUAL APLICADO**

**TEXTO**

**Cambia el color del texto.** <h2 style="color: blue;">CatPhotoApp</h2>

**Cambia el tamaño de la fuente.** font-size: 30px;

**Establece la familia de fuentes de un elemento**. font-family: sans-serif;

**Importa una fuente de Google Fonts**

Para importar una fuente de Google, puedes copiar la URL de la fuente desde la librería de Google Fonts y luego pegarla en tu código HTML. Para este desafío, importaremos la fuente Lobster. Para ello, copia el siguiente fragmento de código y pégalo en la parte superior de tu editor de código (antes de abrir el elemento style):

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

font-family: FAMILY\_NAME, GENERIC\_NAME;

GENERIC\_NAME es opcional, y es el modo de especificar una fuente de reserva o "fallback font" en caso de que la otra fuente especificada no esté disponible. Los nombres de familia de fuentes son sensibles a mayúsculas y minúsculas, y si incluyen espacios deben estar envueltos entre comillas.

**Especifica cómo deben degradarse las fuentes**. Por ejemplo, si quieres que un elemento utilice la fuente Helvetica pero que degrade a la fuente sans-serif en caso que Helvetica no esté disponible, se especifica de la siguiente manera:

font-family: Helvetica, sans-serif;

Los nombres de las fuentes genéricas no distinguen entre mayúsculas y minúsculas. Además, no necesitan comillas porque son palabras clave CSS.

**IMÁGENES**

**Cambia el tamaño de tus imágenes.** CSS tiene una propiedad llamada width que controla el ancho de un elemento. Al igual que con las fuentes, usaremos la unidad de medida px (píxeles) para especificar el ancho de la imagen. Por ejemplo, si queremos crear una clase CSS llamada larger-image que le asigne a los elementos HTML un ancho de 500 píxeles, usamos el siguiente código:

<style>

.larger-image {

width: 500px;

}

</style>

**Añade bordes alrededor de tus elementos**.

Los bordes CSS tienen propiedades como style (estilo), color y width (ancho).

<style>

.thin-red-border {

border-color: red;

border-width: 5px;

border-style: solid;

}

</style>

Recuerda que puedes aplicar múltiples clases a un elemento usando su atributo class, separando cada nombre de clase con un espacio. Por ejemplo:

<img class="class1 class2">

Añade esquinas redondeadas usando border-radius. Ej border-radius: 10px

Haz imágenes circulares usando border-radius de 50%

**Establece el color de fondo de un elemento div.** Puedes establecer el color de fondo de un elemento con la propiedad background-color.

Además de las clases, cada elemento HTML también puede tener un atributo id. Los atributos id deben ser únicos. Los navegadores no verifican esta regla, pero es una buena práctica ampliamente aceptada. Entonces, por favor ten cuidado de no asignar a más de un elemento el mismo atributo id.

<h2 id="cat-photo-app">

Ajusta el padding (relleno) de un elemento. Quizás ya te hayas dado cuenta, pero todos los elementos HTML son, esencialmente, pequeños rectángulos.

Hay tres propiedades importantes que controlan el espacio que rodea cada elemento HTML: padding (relleno), border (borde) y margin (margen).

El padding o relleno de un elemento controla la cantidad de espacio entre su contenido y su border.

Aquí podemos ver que la caja azul y la caja roja están anidadas dentro de la caja amarilla. Fíjate que la caja roja tiene más padding (relleno) que la caja azul.

Cuando aumentas el padding de la caja azul, esto aumenta la distancia (padding) entre el texto y el borde que lo rodea.



El margin (margen) de un elemento controla la cantidad de espacio entre su border y los elementos que lo rodean.

Añade un margen negativo a un elemento. Si estableces el margin de un elemento a un valor negativo, el elemento crecerá de tamaño.

**Añade un "padding" o relleno diferente a cada lado de un elemento.** En ocasiones, querrás personalizar un elemento para que tenga un padding o relleno diferente en cada uno de sus lados. CSS te permite controlar por separado el padding de los cuatro lados individuales de un elemento por medio de las propiedades padding-top, padding-right, padding-bottom y padding-left.

**Utiliza clockwise notation para especificar el relleno (padding) de un elemento.** En lugar de especificar las propiedades padding-top, padding-right, padding-bottom, y padding-left individualmente, puedes especificarlas todas en una sola línea, como se muestra a continuación:

padding: 10px 20px 10px 20px;

**Añade márgenes diferentes a cada lado de un elemento**. En ocasiones, querrás personalizar un elemento para que tenga un margin diferente en cada uno de sus lados. CSS te permite controlar por separado el margin de los cuatro lados individuales de un elemento por medio de las propiedades **margin-top, margin-right, margin-bottom y margin-left.**

**Utiliza clockwise notation para especificar el margen de un elemento.** En lugar de especificar las propiedades margin-top, margin-right, margin-bottom, y margin-left individualmente, puedes especificarlas todas en una sola línea, como se muestra a continuación:

margin: 10px 20px 10px 20px;

**Usa selectores de atributos para dar estilo a elementos**. Hasta ahora, has añadido atributos id o class a elementos para aplicarles estilos específicos. Estos se conocen también como selectores de ID y de clase. Sin embargo, existen otros selectores CSS que puedes utilizar para seleccionar grupos personalizados de elementos a los que quieras aplicar el mismo estilo. [attr=value]. Por ejemplo, el código a continuación cambia los márgenes de todos los elementos que tengan el atributo type con el valor radio:

[type='radio'] {

margin: 20px 0px 20px 0px;

}

**UNIDADES DE MEDIDA**

Varios de los últimos desafíos establecen el "margin" o "padding" de un elemento usando píxeles (px). Los píxeles son un tipo de unidad de longitud que le indica al navegador qué tamaño o cuánto espaciado asignarle a un elemento. Además de px, CSS cuenta con variedad de opciones de unidades de longitud que puedes utilizar.

Los dos tipos principales de unidades de longitud son las unidades absolutas y relativas. Las unidades absolutas están relacionadas con unidades físicas de longitud. Por ejemplo, in y mm se refieren a pulgadas y milímetros, respectivamente. Las unidades de longitud absoluta aproximan la medición real sobre una pantalla, pero existen cierta variación que depende de la resolución de la pantalla utilizada.

Las unidades relativas, como em o rem son relativas a otro valor de longitud. Por ejemplo, em se basa en el tamaño de fuente de un elemento. Si la utilizas para establecer la propiedad font-size, es relativa al font-size del elemento padre.

Nota: Hay varias opciones de unidades relativas que están vinculadas al tamaño del viewport. Veremos estas unidades relativas de medida en la sección de principios de diseño web responsivo.

Aplica un estilo al elemento HTML body. Toda página HTML tiene un elemento body.

Hereda estilos del elemento body. Puedes dar estilo a tu elemento body como a cualquier otro elemento HTML, y todos los demás elementos heredarán los estilos del elemento body.

**Prioriza un estilo por sobre otro.** A veces los elementos HTML reciben múltiples estilos que entran en conflicto entre sí. Lo importante es el orden de las declaraciones de class clases en la sección <style>. La segunda declaración siempre tendrá prioridad sobre la primera.

OJO. El atributo id tiene precedencia sobre el atributo class.

**Sobreescribe declaraciones de clase con inline styles (estilos en línea**). Hemos demostrado que las declaraciones de id tienen prioridad por sobre las declaraciones de clase, independientemente de dónde hayan sido declaradas en el código CSS del elemento style.

Pero existen otras formas de sobreescribir código CSS. ¿Recuerdas los "inline styles" (estilos en línea)?

Usa un inline style para hacer que nuestro elemento h1 sea de color blanco. Recuerda, los estilos en línea se ven así:

<h1 style="color: green;">

**Sobreescribe todos los demás estilos usando !important** Existe una última forma de sobreescribir CSS. Este es el método más poderoso de todos. Pero antes de utilizarlo, consideremos por qué querrías sobreescribir una regla CSS.

En muchas situaciones usarás librerías de CSS. Estas librerías pueden sobreescribir accidentalmente tu propio código CSS. Entonces, cuando necesites asegurarte de que a un elemento se le aplique un código CSS específico, puedes usar !important.

color: red **!important**;

**COLORES**

**Utiliza código hexadecimal (hex code) para indicar colores específico**. En CSS, podemos representar colores utilizando 6 dígitos hexadecimales, dos para los componentes de rojo (R), verde (G) y azul (B). Por ejemplo, #000000 corresponde al color negro, y también es el valor más bajo posible. Puedes encontrar más información sobre el [sistema de colores RGB aquí](https://www.freecodecamp.org/news/rgb-color-html-and-css-guide/#whatisthergbcolormodel).

Afortunadamente, puedes abreviar gran parte de ellos.

Color Hex code abreviado

Cyan (cian) #0FF

Green (verde) #0F0

Red (rojo) #F00

Fuchsia (fucsia) #F0F

Utiliza valores RGB para asignar color a los elementos. Otra forma de representar colores en CSS es utilizar valores RGB. El valor RGB del color negro se ve así: rgb(255, 255, 255)

Color RGB

Blue (azul) rgb(0, 0, 255)

Red (rojo) rgb(255, 0, 0)

Orchid (color orquídea) rgb(218, 112, 214)

Sienna (siena) rgb(160, 82, 45)

**VARIABLES**

**Usa variables CSS para modificar varios elementos a la vez.** Las Variables CSS son una manera poderosa de modificar varias propiedades de estilos CSS a la vez, cambiando su valor en un único sitio.

Para crear una variable CSS, solo tienes que darle un nombre que comience con dos guiones (--) y asignarle un valor, como se muestra a continuación:

--penguin-skin: gray;

Para crear una variable CSS, solo tienes que darle un nombre que comience con dos guiones (--) y asignarle un valor, como se muestra a continuación:

background: var(--penguin-skin);

Agrega un valor de respaldo (fallback) a una variable CSS. Cuando utilices tu variable como valor de una propiedad CSS, puedes adjuntar un valor de respaldo o "fallback", que será utilizado por el navegador si la variable dada no es válida.

background: var(--penguin-skin, black);

Cuando creas una variable, queda disponible para que la utilices dentro del selector en el que la hayas creado. Esa variable también estará disponible en cualquiera de los descendientes de ese selector. Esto ocurre porque las variables CSS son heredadas, al igual que las propiedades comunes.

Para hacer uso de la herencia, las variables CSS suelen ser definidas en el elemento :root.

:root es un "pseudo-class selector" (selector de pseudo-clase) que corresponde al elemento raíz o "root" del documento, que generalmente es el elemento html. Al crear tus variables en :root, estarán disponibles globalmente y se podrán acceder desde cualquier otro selector en la hoja de estilo.

**MEDIA QUERY (CELULARES)**

Las variables CSS pueden simplificar la forma en que utilizas "media queries" (consultas sobre el tipo de dispositivo donde se muestra el documento HTML).

Por ejemplo, cuando la pantalla es más pequeña o más grande que el breakpoint de tu media query, puedes cambiar el valor de una variable, y su estilo se aplicará dondequiera que la utilices.

  @media (max-width: 350px) {

    :root {

      /\* Cambia solo el código debajo de esta línea \*/

    --penguin-size: 200px;

    --penguin-skin: black;

      /\* Cambia solo el código encima de esta línea \*/

    }

  }

**DISEÑO VISUAL APLICADO**

El texto es frecuentemente una gran parte del contenido web. CSS tiene múltiples opciones para alinearlo con la propiedad text-align.

text-align: justify; espacia el texto para que cada línea tenga el mismo ancho.

text-align: center; centra el texto

text-align: right; alinea el texto hacia la derecha

Y text-align: left; (opción por defecto) alinea el texto hacia la izquierda.

Puedes especificar la altura de un elemento con la propiedad **height** en CSS como lo haces con la propiedad **width** (ancho).

Utiliza la etiqueta **strong** para poner el texto en negrita. Esto es equivalente a darle un estilo font-weight: bold;

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

Para subrayar texto, puedes usar la etiqueta **u**. Esto se utiliza a menudo para indicar que una sección del texto es importante, o algo que hay que recordar. Con la etiqueta u, el navegador aplica el CSS de text-decoration: underline; al elemento.

Para enfatizar el texto, puedes usar la etiqueta **em**. Esto muestra el texto en cursiva, ya que el navegador aplica el CSS de font-style: italic; al elemento.

Para tachar el texto, que es cuando una línea horizontal atraviesa los caracteres, puede usar la etiqueta **s**. Muestra que una sección de texto ya no es válida. Con la etiqueta s, el navegador aplica el CSS de text-decoration: line-through; al elemento.

Puedes usar la etiqueta **hr** para agregar una línea horizontal a través del ancho de su elemento contenedor. Esto se puede usar para definir un cambio de tema o para separa grupos de contenido visualmente. En HTML, la etiqueta hr se cierra sola, por lo tanto no necesita una etiqueta de cierre por separado.

Ajusta la propiedad background-color del texto. En lugar de ajustar el fondo general o el color del texto para que el primer plano sea fácilmente legible, puedes agregar un background-color al elemento que contiene el texto que deseas destacar. Este reto utiliza rgba() en lugar de códigos hex o rgb() normal.

rgba significa:

r = red

g = green

b = blue

a = alfa/nivel de opacidad

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

**Agrega box-shadow a un elemento tipo tarjeta**. La propiedad box-shadow aplica una o más sombras a un elemento. La propiedad box-shadow toma los siguientes valores, en orden:

offset-x (qué tan lejos extender la sombra horizontalmente desde el elemento)

offset-y (qué tan lejos extender la sombra verticalmente desde el elemento)

blur-radius

spread-radius

color

Los valores blur-radius y spread-radius son opcionales.

Se pueden crear múltiples box-shadows usando comas para separar las propiedades de cada elemento box-shadow.

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

Disminuir la opacidad de un elemento. La propiedad opacity en CSS se usa para ajustar la opacidad o, por el contrario, la transparencia de un elemento.

Un valor de 1 es opaco, que no es transparente en absoluto.

Un valor de 0.5 es la mitad transparente.

Un valor de 0 es completamente transparente.

**Usa la propiedad text-transform para hacer el texto mayúsculas.** La propiedad text-transform en CSS se utiliza para cambiar la apariencia del texto. Es una forma conveniente de asegurarse de que el texto en una página web aparezca de manera consistente, sin tener que cambiar el contenido del texto de los elementos HTML reales.

La siguiente tabla muestra como los diferentes valores de text-transform cambian el texto de ejemplo "Transformame".

Valor Resultado

lowercase "Transformame"

uppercase "TRANSFORMAME"

capitalize "Transformame"

initial Usa el valor predeterminado

inherit Utiliza el valor text-transform del elemento principal

none Predeterminado: Usa el texto original

**Establece el font-weight para varios elementos de títulos**. La propiedad font-weight establece que tan gruesos o delgados son los caracteres en una sección de texto.

**Establece la line-height de los párrafos**. CSS ofrece la propiedad line-height para cambiar la altura de cada línea en un bloque de texto. Como sugiere el nombre, cambia la cantidad de espacio vertical que recibe cada línea de texto.

**Ajusta el hover de una etiqueta anchor.** Una pseudo-clase es una palabra clave que se puede agregar a los selectores para seleccionar un estado específico de un elemento.

Por ejemplo, el estilo de una etiqueta "anchor" puede ser cambiado por el estado de su hover utilizando el selector de pseudo-clase :hover. Aquí está el CSS para cambiar el color color de la etiqueta de "anchor" a rojo durante el estado hover: (Hover es cuando me paro encima del elemento con el puntero del mouse).

a:hover {

color: red;

}

**POSICIONAMIENTO DE LOS ELEMENTOS**

CSS trata cada elemento HTML como su propia caja, esto es a lo que usualmente se refiere como el Modelo de Caja de CSS. Los elementos bloque automáticamente empiezan en una nueva línea (piensa en las etiquetas título, párrafos y divs) mientras que los elementos en línea se ubican entre el contenido al rededor (como imágenes o spans). El diseño por defecto de los elementos en esta manera se llama el flujo normal de un documento, pero CSS ofrece la propiedad position para sobreescribirlo.

Cuando la posición de un elemento se establece a relative, te permite especificar como CSS lo moverá relativo a su posición actual dentro del flujo normal de la página. Se empareja con las propiedades de desplazamiento CSS de left o right, y top o bottom. Estas dicen cuántos pixeles, porcentajes, o ems se debe mover el elemento lejos de donde esté normalmente posicionado. El siguiente ejemplo mueve el párrafo 10 pixeles lejos de la parte inferior:

p {

position: relative;

bottom: 10px;

}

**Bloquea un elemento con relación a su padre con el posicionamiento absoluto.** La siguiente opción para la propiedad CSS position es absolute, que bloquea el elemento en su lugar en relación con su contenedor principal. A diferencia de la posición relative, esto elimina el elemento del flujo normal del documento, por lo que los elementos circundantes lo ignoran. Las propiedades de desplazamiento de CSS (superior o inferior e izquierda o derecha) se utilizan para ajustar la posición.

Un matiz del posicionamiento absoluto es que estará bloqueado en relación con su antepasado posicionado más cercano. Si olvidas agregar una regla de posición al elemento principal, (esto generalmente se hace usando position: relative;), el navegador seguirá buscando en la jerarquía de elementos y, en última instancia tomará por defecto la etiqueta body.

  #searchbar {

    position: absolute;

    top: 50px;

    right: 50px;

  }

Bloquea un elemento a la ventana del navegador con el posicionamiento fijo. El siguiente esquema de diseño que ofrece CSS es la posición fixed, que es un tipo de posicionamiento absoluto que bloquea un elemento relativo a la ventana del navegador. Similar al posicionamiento absoluto, se usa con las propiedades de desplazamiento CSS y también elimina el elemento del flujo normal del documento. Otros elementos ya no "se dan cuenta" de donde se coloca, lo que puede requerir algunos ajustes de diseño en otros lugares.

Una diferencia clave entre las posiciones fixed y absolute es que un elemento con una posición fija (fixed) no se moverá cuando el usuario se desplace.

**Empuja elementos hacia la izquierda o hacia la derecha con la propiedad** **float** La siguiente herramienta de posicionamiento en realidad no usa position, sino que establece la propiedad float de un elemento. Los elementos flotantes se eliminan del flujo normal de un documento y se empujan a left o right de su elemento padre contenedor. Se usa comúnmente con la propiedad width para especificar cuanto espacio horizontal requiere el elemento flotante.

**Cambia la posición de los elementos superpuestos con la propiedad z-index**. Cuando los elementos son posicionados para superponerse (i.e. usando position: absolute | relative | fixed | sticky), el elemento que viene después dentro del marcado HTML aparecerá, por defecto, encima de los otros elementos. Sin embargo, la propiedad z-index puede especificar el orden de cómo los elementos están apilados unos sobre otros. Debe ser un entero (i.e. un número entero y no un decimal), y los elementos que mayor valor tengan en z-index serán movidos más arriba en la pila de elementos que aquellos con valores menores.

**Centra un elemento horizontalmente usando la propiedad margin**. Otra técnica de posicionamiento consiste en centrar un elemento de bloque horizontalmente. Una manera de hacer esto es que margin tenga valor auto.

**Este método también funciona para imágenes**. Las imágenes son elementos en línea de forma predeterminada, pero se pueden cambiar a elementos de bloque cuando se establece la propiedad display en block.

**Centra el div en la página agregando una propiedad margin con un valor de auto.**

**COLORES**

Me salte algunos, revisar

Crea un gradiente lineal de CSS gradual. La aplicación de un color en elementos HTML no se limita a un tono plano. CSS proporciona la capacidad de usar transiciones de color, también conocidas como degradados, en los elementos. Esto se accede a través de la función linear-gradient() de la propiedad background. Aquí está la sintaxis general:

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

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

Crea textura agregando un patrón sutil como imagen de fondo. Una forma de agregar textura e interés a un fondo y hacer que se destaque más es agregar un patrón sutil. La clave está en el balance, dado que no querrás que el fondo destaque demasiado y quite atención al primer plano. La propiedad background acepta la función url() para enlazar una imagen de la textura o patrón elegido. El enlace es cubierto entre comillas dentro del paréntesis.

Utiliza la propiedad de escala de transformación CSS para cambiar el tamaño de un elemento. Para cambiar la escala de un elemento, CSS tiene la propiedad transform, junto con su función scale(). En el ejemplo de código siguiente se duplica el tamaño de todos los elementos de párrafo de la página:

p {

transform: scale(2);

}

Utiliza la propiedad de escala de transformación CSS para escalar un elemento al desplazarse. La propiedad transform tiene una variedad de funciones que te permiten escalar, mover, rotar, sesgar, etc., sus elementos. Cuando se usa con pseudo-classes como :hover que especifican un cierto estado de un elemento, la propiedad transform puede agregar fácilmente interactividad a sus elementos.

**KEYFRAMES Y ANIMACION**

Para animar un elemento, necesitas conocer las propiedades de animación y la regla @keyframes. Las propiedades de animación controlan como debe comportarse la animación y la regla @keyframes controla lo que sucede durante esa animación. Hay ocho propiedades de animación en total. Este desafío lo mantendrá simple y cubrirá primero los dos más importantes:

animation-name establece el nombre de la animación, que luego es utilizada por @keyframes para decirle a CSS que reglas van con que animaciones.

animation-duration establece el tiempo de la animación.

@keyframes es como especificar exactamente lo que sucede dentro de la animación durante la duración. Esto se hace dando propiedades CSS para "marcos" específicos durante la animación, con porcentajes que van del 0% al 100%. Si comparas esto con una película, las propiedades de CSS de 0% es como se muestra el elemento en la escena inicial. Las propiedades de CSS con 100% es como aparece el elemento al final, justo antes de que rueden los créditos. Luego, CSS aplica la magia para hacer la transición del elemento durante la duración dada para representar la escena. Aquí hay un ejemplo para ilustrar el uso de @keyframes y las propiedades de animación:

#anim {

animation-name: colorful;

animation-duration: 3s;

}

@keyframes colorful {

0% {

background-color: blue;

}

100% {

background-color: yellow;

}

}

Para el elemento anim id, el fragmento de codigo anterior establece el animation-name para colorful y establece el animation-duration a 3 segundos. A continuación, la regla @keyframes vincula a las propiedades de animación con el nombre colorful. Establece el color en azul al principio de la animación (0%) que pasara a amarillo al final de la animación (100%). No estás limitado solo a las transiciones de principio a fin, puedes establecer propiedades para el elemento para cualquier porcentaje entre 0% y 100%

Modifica el modo de relleno de una animación (animation-fill-mode). Eso es genial, pero aún no funciona bien. Observa como la animación se restablece después de que haya pasado 500ms, haciendo que el botón vuelva al color original. Lo que quieres es que el botón permanezca resaltado.

Esto se puede hacer estableciendo la propiedad animation-fill-mode en forwards. El animation-fill-mode especifica el estilo aplicado a un elemento cuando la animación ha finalizado. Puedes configurarlo así: animation-fill-mode: forwards;

<style>

  button {

    border-radius: 5px;

    color: white;

    background-color: #0F5897;

    padding: 5px 10px 8px 10px;

  }

  button:hover {

    animation-name: background-color;

    animation-duration: 500ms;

    /\* Cambia solo el código debajo de esta línea \*/

    animation-fill-mode: forwards;

    /\* Cambia solo el código encima de esta línea \*/

  }

  @keyframes background-color {

    100% {

      background-color: #4791d0;

    }

  }

</style>

<button>Register</button>

**Crear movimiento usando animación CSS**. Cuando los elementos tienen una position, como fixed o relative, las propiedades de desplazamiento CSS right, left, top y bottom se pueden usar en las reglas de animación para crear movimiento.

Como se muestra en el siguiente ejemplo, puedes empujar el elemento hacia abajo y luego hacia arriba estableciendo la propiedad top fotograma clave (keyframe) a 50% en 50px, pero estableciéndolo en 0px para el primer fotograma clave (0%) y el último a (100%).

<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;

      top: 0px;

      left: 0px;

    }

    50% {

      background-color: green;

      top: 50px;

      left: 25px;

    }

    100% {

      background-color: yellow;

      top: 0px;

      left: -25px;

    }

  }

</style>

<div id="rect"></div>

Crear dirección visual desvaneciendo un elemento de izquierda a derecha. Para este desafío, cambiarás la opacidad (opacity) de un elemento animado para que se desvanezca gradualmente a medida que llega al lado derecho de la pantalla.

En la animación mostrada, el elemento redondo con el fondo degradado se mueve hacia la derecha con la marca del 50% de la animación según la regla @keyframes.

**Animar los elementos continuamente utilizando un contador de animaciones infinitas.**

En los desafíos anteriores, vimos cómo utilizar algunas de las propiedades de la animación y la regla @keyframes. Otra propiedad de animación es la animation-iteration-count la cual te permite controlar cuántas veces te gustaría hacer un bucle a través de la animación

    animation-name: bounce;

    animation-duration: 1s;

    animation-iteration-count: infinite;

Corazon latiendo:

<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>

**CCS FLEXBOX**

Colocar la propiedad CSS display: flex; en un elemento te permite usar otras propiedades flex para construir una página responsiva.

Utiliza la propiedad flex-direction para hacer una fila. Agregando display: flex a un elemento lo convierte en un contenedor flexible. Esto permite alinear cualquier elemento secundario de ese elemento en filas o columnas. Para ello, agrega la propiedad flex-direction al elemento principal y configúralo en fila o columna. La creación de una fila alineara los elementos secundarios horizontalmente, y la creación de una columna alineara los elementos secundarios verticalmente.

Otras opciones para flex-direction son row-reverse y column-reverse.

Nota: El valor predeterminado para la propiedad flex-direction es row.

**Alinea elementos mediante la propiedad justify-content.** Algunas veces los elementos flexibles dentro de un contenedor flexible no llenan todo el espacio del contenedor. Es común querer indicarle al CSS cómo alinear y espaciar los elementos flexibles de una determinada manera. Afortunadamente, la propiedad justify-content tiene varias opciones para hacer esto. Pero primero, hay que entender alguna terminología importante antes de revisar dichas opciones.

Para más información acerca de las propiedades de flex-box

Recuerda que establecer un contenedor flexible como fila coloca los elementos flexibles uno al lado del otro de izquierda a derecha. Un contenedor flexible establecido como columna coloca los elementos flexibles apilados verticalmente de arriba a abajo. Para cada uno, la dirección en la que están dispuestos los elementos flexibles se llama el eje principal. Para una fila, esta es una línea horizontal que recorta cada elemento. Y para una columna, el eje principal es una línea vertical a través de los elementos.

**Hay varias opciones para espaciar los elementos flexibles a lo largo de la línea que representa el eje principal. Uno de los más utilizados es justify-content: center;, el cual alinea hacia el centro todos los elementos flexibles dentro del contenedor flexible. Otras opciones incluyen:**

* flex-start: alinea los elementos con el inicio del contenedor flex. Para una fila, esto empuja los elementos a la izquierda del contenedor. Para una columna, esto empuja los elementos a la parte superior del contenedor. Esta es la alineación predeterminada si no se especifica ningún tipo de justify-content.
* flex-end: alinea los elementos con el final del contenedor flex. Para una fila, esto empuja los elementos a la derecha del contenedor. Para una columna, esto empuja los elementos a la parte inferior del contenedor.
* space-between: alinea los elementos en el centro del eje principal, con un espacio extra entre los elementos. Los primeros y últimos elementos son empujados hasta el borde del contenedor flexible. Por ejemplo, en una fila el primer elemento está en el lado izquierdo del contenedor, el último elemento está en el lado derecho del contenedor, luego el espacio restante se distribuye uniformemente entre los demás elementos.
* space-around: similar a space-between pero los primeros y últimos elementos no están fijados en los bordes del contenedor, el espacio se distribuye alrededor de todos los elementos con la mitad de un espacio en ambos extremos del contenedor flexible.
* space-evenly: Distribuye el espacio de manera uniforme entre los flex items con un espacio completo en cualquier extremo del flex container.

**Alinea elementos mediante la propiedad align-items.** La propiedad align-items es similar a justify-content. Recuerda que la propiedad justify-content alineó los elementos flexibles a lo largo del eje principal. Para las filas, el eje principal es una línea horizontal y para las columnas es una vertical.

Los contenedores flexibles también tienen un eje transversal que es el opuesto al eje principal. Para las filas, el eje transversal es vertical y para las columnas, el eje transversal es horizontal.

**CSS ofrece la propiedad align-items para alinear elementos flexibles a lo largo del eje transversal. Para una fila, le indica al CSS como empujar los elementos en toda la fila hacia arriba o hacia abajo dentro del contenedor. Y para una columna, como empujar todos los elementos hacia la izquierda o hacia la derecha dentro del contenedor.**

Los diferentes valores disponibles para align-items incluyen:

* flex-start: alinea los elementos con el inicio del contenedor flexible. Para las filas, esto alinea los elementos a la parte superior del contenedor. Para las columnas, esto alinea los elementos a la parte izquierda del contenedor.
* flex-end: alinea los elementos con el final del contenedor flexible. Para las filas, esto alinea los elementos a la parte inferior del contenedor. Para las columnas, esto alinea los elementos a la parte derecha del contenedor.
* center: alinea los elementos hacia el centro. Para las filas, esto alinea los elementos verticalmente (igual espacio por encima y por debajo de los elementos). Para columnas, esto las alinea horizontalmente (igual espacio a la izquierda y a la derecha de los elementos).
* stretch: estira los elementos para llenar el contenedor flexible. Por ejemplo, los elementos de filas son estirados para llenar el contenedor flexible de arriba hacia abajo. Este es el valor predeterminado si no se especifica ningún tipo de align-items.
* baseline: alinea los elementos con sus líneas base. Una línea base es un concepto de texto, piensa en ella como la línea en la que se sitúan las letras.

**Usa la propiedad flex-wrap para envolver una fila o columna.**

CSS flexbox tiene una característica para dividir un flex container en varias filas (o columnas). De forma predeterminada, un contenedor flexible encajará todos los elementos flexibles juntos. Por ejemplo, una fila estará completa en una sola línea.

Sin embargo, usar la propiedad flex-wrap le indica al CSS que envuelva los elementos. Esto significa que los elementos extra se mueven hacia una nueva fila o columna. El punto de ruptura donde ocurre la envoltura depende del tamaño de los elementos y del tamaño del contenedor.

CSS también tiene opciones para la dirección de la envoltura:

nowrap: esta es la configuración predeterminada, y no envuelve elementos.

wrap: envuelve elementos en múltiples líneas de arriba a abajo si están en filas y de izquierda a derecha si están en columnas.

wrap-reverse: envuelve elementos en múltiples líneas de abajo hacia arriba si están en filas y de derecha a izquierda si están en columnas.

**Utiliza la propiedad flex-shrink para reducir elementos**. Hasta ahora, todas las propiedades en los desafíos se aplican al contenedor flexible (el padre de los elementos flex). Sin embargo, hay varias propiedades útiles para los elementos flex.

La primera es la propiedad flex-shrink. Cuando se usa, permite que un elemento se contraiga si el contenedor flex es demasiado pequeño. Los elementos se reducen cuando el ancho del contenedor principal es menor que el ancho combinado de todos los elementos flex dentro del él.

La propiedad flex-shrink toma números como valores. Cuando mayor sea el número, más se reducirá en comparación con los otros elementos en el contenedor. Por ejemplo, si un elemento tiene un flex-shrink con valor de 1 y el otro tiene un flex-shrink con valor de 3, el que tiene el valor de 3 se reducirá tres veces más que el otro.

**Usa la propiedad flex-grow para expandir elementos**. Lo contrario de flex-shrink es la propiedad flex-grow. Recuerda que flex-shrink controla el tamaño de los elementos cuando el contenedor se encoge. La propiedad flex-grow controla el tamaño de los elementos cuando el contenedor primario se expande.

Utilizando un ejemplo similar al del último desafío, si un elemento tiene un flex-grow con valor de 1 y el otro tiene un flex-grow con valor de 3, el que tiene el valor de 3 crecerá tres veces más que el otro.

Le puedo dar 3 propiedades de una a border

border: '5px solid pink',