Tabla de contenido

[HTML Básico]

{Lección 1 y 2} Títulos (Headlines):

Este elemento le informa al navegador sobre la estructura de tu sitio web.

<h1> Para título Principal </h1>

<h2> Para Subtítulos (h2,h3,h4...) </h2>

{Lección 3} Párrafos (paragraph):

Los elementos **p** son el elemento preferido para el texto de los párrafos en los sitios web.

* Texto del Parrafo *

{Lección 4} Relleno de Espacio con Texto Random :

* lorem ipsum *

{Lección 5 y 6} Comentar en Html:

<!-- Comentario ó Codigo que quiere desactivarse momentaneamente -->

{Lección 8} Descripción de Elementos HTML5:

HTML5 introduce etiquetas HTML más descriptivas. Estas incluyen main, header, footer, nav, video, article, section, entre otras.

Estas etiquetas dan una estructura descriptiva a tu HTML, hacen que tu HTML sea más fácil de leer, ayudan con la Optimización de Motores de Búsqueda (SEO) y mejoran la accesibilidad.

{Lección 8} Elemento main (principal) html5 :

Ayuda a los motores de búsqueda y otros desarrolladores a encontrar el contenido principal de tu página.

<main>...contenido principal de la página...</main>

{Lección 9} Agregar Imagen al sitio Web:

Puedes agregar imágenes a tu sitio web utilizando el elemento img, y apuntar a la URL de una imagen específica usando el atributo src.

Todos los elementos img deben tener un atributo alt. El texto dentro de un atributo alt es utilizado por los lectores de pantalla para mejorar la accesibilidad y se muestra si la imagen falla en cargar.

Note: Si la imagen es puramente decorativa, usar un atributo alt vacío es una buena práctica.

<img src="https://www.Imagen Web.com/your-image.jpg" alt="Texto descriptivo alternativo">

{Lección 10} Enlace hacia una pagina externa con el elemento a (anchor):

Los elementos a requieren un atributo href con la dirección web de destino. También necesitan un texto anchor.

* Texto anchor que direcciona a la pagina externa *

target es un atributo opcional de etiqueta anchor que especifica dónde abrir el enlace. El valor _blank especifica abrir el enlace en una nueva pestaña.

{Lección 11} Enlace hacia sección interna con el elemento a (anchor):

Para crear un enlace interno, asignas el atributo href de un enlace con un símbolo hash # más el valor del atributo id para el elemento al que deseas enlazar internamente, normalmente más abajo de la página. Luego necesitas agregar el mismo atributo id al elemento que estás enlazando. Un id es un atributo que describe un elemento de forma única.

* Texto que envia a la sección *

<h2 id="nombre que idetifica el elemento"> Elemento que se quiere enlazar </h2>

{Lección 12} Anidar un elemento anchor dentro de un párrafo:

Puedes anidar enlaces dentro de otros elementos de texto.

<p>

Here's a link to freecodecamp.org for you to follow.

</p>

El texto, link to freecodecamp.org, dentro de un elemento a se llama anchor text, y mostrará el enlace para hacer clic

{Lección 13} Crear un enlace muerto :

A veces quieres agregar elementos a en tu sitio web antes de saber dónde se enlazarán, por ello se puede crear un enlace muerto asignando el valor de href con #.

cat photos

{Lección 14} Convertir una imagen en un enlace:

Puedes convertir elementos en enlaces, anidándolos dentro de un elemento a.

<img src="https://bit.ly/fcc-running-cats" alt="Three kittens running towards the camera.">

{Lección 15} Crear una lista no ordenada (Unordered List):

Las listas no ordenadas comienzan con un elemento <ul> de apertura, seguido de cualquier número de elementos <li>.

```
<ul>
  <li>milk</li>
  <li>cheese</li>
</ul>
```

{Lección 16} Crear una lista ordenada o numerada (Ordered List):

Las listas ordenadas comienzan con un elemento de apertura <ol>, seguido de cualquier número de elementos <li>.

```
<ol>
  <li>Garfield</li>
  <li>Sylvester</li>
</ol>
```

{Lección 17} Crear un campo de Texto o formulario:

Los elementos de entrada input son una forma conveniente de obtener información de tu usuario.

Nota: Los elementos input se cierran solos.

```
<input type="text">
```

{Lección 18} Agregar texto provisional a un campo de texto:

El texto provisional es lo que se muestra en tu elemento de entrada input antes de que el usuario haya ingresado nada.

```
<input type="text" placeholder="this is placeholder text">
```

{Lección 19} Crear un elemento formulario:

Puedes construir formularios web que realmente envíen datos a un servidor usando sólo HTML puro. Puedes hacer esto especificando un atributo action en tu elemento form.

```
<form action="/url-where-you-want-to-submit-form-data">
  <input ...>
</form>
```

{Lección 20} Agregar un botón de envío a un formulario:

Agreguemos un botón ***submit*** a tu formulario. Al hacer clic en este botón se enviarán los datos de tu formulario a la URL especificada con el atributo action de tu formulario.

```
<button type="submit">this button submits the form</button>
```

{Lección 21} Uso de HTML5 para requerir un campo:

Puedes requerir campos específicos de un formulario para que tu usuario no pueda enviarlo hasta que no los haya rellenado. Si deseas hacer obligatorio un campo de entrada de texto, puedes agregar el atributo required dentro de tu elemento input

```
<input type="text" required>
```

{Lección 22} Crear un conjunto de botones de radio (radio buttons):

Puedes usar botones de radio para preguntas en las que quieres que el usuario solo te dé una respuesta a partir de múltiples opciones.

Los botones de radio son un tipo de entrada input.

Cada uno de tus botones de radio puede anidarse dentro de su propio elemento label. Envolver un elemento input dentro de un elemento label asociará automáticamente la entrada del botón de radio con el elemento label que lo rodea.

Todos los botones de radio relacionados deben tener el mismo atributo name para crear un grupo de botones de radio. Al crear un grupo de radio, si se selecciona cualquier botón de radio se deselecciona automáticamente los otros botones dentro del mismo grupo, asegurándose que el usuario proporcione solo una respuesta.

```
<label>
  <input type="radio" name="indoor-outdoor">Indoor
</label>
```

Nota: Se considera buena práctica establecer un atributo for en el elemento label, con un valor que coincida con el valor del atributo id del elemento input. Esto permite a las tecnologías asistivas crear una relación vinculada entre la etiqueta y el elemento hijo input.

{Lección 23} Crea un conjunto de casillas de verificación (checkbox):

Los formularios suelen usar casillas de verificación (checkboxes) para preguntas que puedan tener más de una respuesta.

Las casillas de verificación son un tipo de input.

Cada una de tus casillas de verificación puede anidarse dentro de su propio elemento label.

Todos los inputs de tipo casilla de verificación que están relacionados entre sí deben tener el mismo atributo name.

Se considera buena práctica definir explícitamente la relación entre un input de tipo checkbox (casilla de verificación) y su correspondiente label (etiqueta), estableciendo el atributo for en el elemento label para que coincida con el atributo id del input asociado.

<label for="loving"><input id="loving" type="checkbox" name="personality"> Loving</label>

{Lección 24} Usar el atributo value con botones de radio y casillas de verificación:

Cuando se envía un formulario, los datos se envían al servidor e incluyen entradas para las opciones seleccionadas. Los inputs de tipo radio y checkbox reportan sus valores desde el atributo value.

<label for="indoor">

<input id="indoor" value="indoor" type="radio" name="indoor-outdoor">Indoor

</label>

<label for="outdoor">

<input id="outdoor" value="outdoor" type="radio" name="indoor-outdoor">Outdoor

</label>

Aquí tienes dos inputs de tipo radio. Cuando el usuario envía el formulario con la opción indoor seleccionada, los datos del formulario incluirán la línea: indoor-outdoor=indoor. Esto proviene de los atributos name y value del input "indoor".

Si omites el atributo value, los datos del formulario enviado utilizarán el valor por defecto, que es on. En este escenario, si el usuario hizo click en la opción "indoor" y envió el formulario, el dato resultante del formulario sería indoor-outdoor=on, lo cual no resulta útil. Por lo que el atributo value debe establecerse a un valor que identifique la opción claramente.

{Lección 25} Marcar botones de radio y casillas de verificación por defecto:

Puedes hacer que una casilla de verificación o botón de radio se marque por defecto usando el atributo checked.

Para hacer esto, simplemente agrega la palabra checked al interior de un elemento de entrada.

<input type="radio" name="test-name" checked>

{Lección 26} Anidar muchos elementos dentro de un solo elemento div:

El elemento div, también conocido como elemento de división, es un contenedor de propósito general para otros elementos.

El elemento div es probablemente el elemento HTML más utilizado de todos.

Al igual que cualquier otro elemento sin cierre automático, puedes abrir un elemento div con * y cerrarlo en otra línea con .*

{Lección 27} Declarar el Doctype de un documento HTML:

Hasta el momento se han cubierto elementos HTML específicos y sus usos. Sin embargo, hay algunos elementos que dan una estructura general a tu página, y deben incluirse en cada documento HTML.

En la parte superior de tu documento, necesitas decirle al navegador qué versión de HTML está utilizando tu página. HTML es un lenguaje en evolución, y se actualiza regularmente. La mayoría de los navegadores principales soportan la última especificación, que es HTML5. Sin embargo, páginas web más antiguas puede que hagan uso de versiones anteriores del lenguaje.

Proporcionas al navegador esta información agregando la etiqueta *<!DOCTYPE ...>* en la primera línea, donde la parte ... es la versión de HTML. Para HTML5, utilizas *<!DOCTYPE html>*.

El ! y DOCTYPE en mayúsculas es importante, especialmente para los navegadores más antiguos. El html no es sensible a mayúsculas y minúsculas.

A continuación, el resto de tu código HTML necesita ser envuelto en etiquetas html. La apertura *<html>* va directamente debajo de la línea *<!DOCTYPE html>,* y el cierre *</html>* va en el final de la página.

```
<!DOCTYPE html>

<html>


</html>
```

{Lección 28} Definir el encabezado y el cuerpo de un Documento HTML:

Puedes agregar otro nivel de organización en tu documento HTML dentro de las etiquetas html con los elementos head y body. Cualquier código con información sobre tu página iría dentro de la etiqueta head. Entonces, cualquier código con el contenido de la página (lo que se muestra para un usuario) iría dentro de la etiqueta body.

Elementos de metadatos, tales como link, meta, title, y style, típicamente van dentro del elemento head.

```
<!DOCTYPE html>
<html>
  <head>
    <meta />
  </head>
  <body>
    <div>
    </div>
  </body>
</html>
```

[CSS Basico]

{Lección 1} Cambiar el color del texto:

Podemos hacer esto cambiando el style de tu elemento (h2 para el ejemplo).

La propiedad que es responsable del color del texto de un elemento es la propiedad de estilo color.

```
<h2 style="color: blue;">CatPhotoApp</h2>
```

Nota: Ten en cuenta que es una buena práctica terminar las declaraciones de inline style usando punto y coma (;).

{Lección 2} Utilizar selectores CSS para dar estilo a elementos:

En CSS tienes a tu disposición cientos de propiedades CSS que puedes usar para cambiar el aspecto de un elemento de tu página.

Cuando escribiste `<h2 style="color: red;">CatPhotoApp</h2>` , estabas aplicando un estilo a ese elemento h2 específico usando CSS en línea.

Inline CSS es una de las formas de especificar el estilo de un elemento, pero existe una mejor forma de aplicar estilos CSS.

En la parte superior de tu código, crea un bloque style.

Dentro de ese bloque de estilo (style), puedes crear un selector CSS para todos los elementos h2. Por ejemplo, para que todos los elementos h2 sean de color rojo puedes añadir una regla de estilo como la siguiente:

```
<style>
  h2 {
    color: red;
```

```
  }
</style>
```

Nota: Ten en cuenta que es importante tener tanto llaves de apertura como de cierre ({ y }) alrededor de la(s) regla(s) de estilo de cada elemento. También debes asegurarte de que tu definición de estilo para ese elemento esté dentro de las etiquetas de apertura y cierre de estilo. Por último, asegúrate de añadir un punto y coma (;) al final de cada una de las reglas de estilo de tu elemento.

{Lección 3} Usar una clase CSS para aplicar estilos a un elemento:

Las clases son estilos reutilizables que se pueden agregar a los elementos HTML.

A continuación te presentamos un ejemplo de cómo declarar una clase CSS

```
<style>
  .blue-text {
    color: blue;
  }
</style>
```

Puedes ver que hemos creado una clase CSS llamada blue-text dentro de la etiqueta *<style>*. Puedes aplicar una clase a un elemento HTML de este modo: *<h2 class="blue-text">CatPhotoApp</h2>*. Ten en cuenta que en tu elemento CSS style, los nombres de clase comienzan con un punto. En el atributo "class" de tus elementos HTML, el nombre de la clase no lleva punto delante.

Nota:

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">
```

{Lección 4} Aplicar un estilo a múltiples elementos usando una clase CSS:

Las clases te permiten usar los mismos estilos CSS en múltiples elementos HTML. Puedes ver esto aplicando tu clase de red-text al primer elemento p.

{Lección 5} Cambia el tamaño de fuente de un elemento:

El tamaño de la fuente es controlado por la propiedad CSS font-size, como se muestra a continuación

```
h1 {
  font-size: 30px;
}
```

{Lección 6} Establece la familia de fuentes de un elemento:

Puedes utilizar la propiedad font-family (familia de fuente) para establecer qué fuente debe usar un elemento.

Por ejemplo, si quieres establecer la fuente de tu elemento h2 a *sans-serif*, usarías el siguiente CSS

```
h2 {
  font-family: sans-serif;
}
```

{Lección 7} Importar una fuente de Google Fonts:

Además de especificar fuentes comunes disponibles en la mayoría de los sistemas operativos, también podemos especificar fuentes web no estándar y personalizadas para usarlas en nuestro sitio web. En Internet hay muchos lugares donde podemos conseguir web fonts. Para este ejemplo nos centraremos en la biblioteca de Google Fonts.

Google Fonts es una biblioteca gratuita de web fonts que puedes utilizar en tu código CSS, haciendo referencia a la URL de la fuente.

Entonces, importemos y apliquemos una fuente de Google (ten en cuenta que si Google está bloqueado en tu país, tendrás que pasar por alto este desafío).

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">

Ahora puedes usar la fuente Lobster en tu CSS usando Lobster como FAMILY_NAME como en el siguiente ejemplo

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. Veremos esto en el siguiente desafío.

Los nombres de familia de fuentes son sensibles a mayúsculas y minúsculas, y si incluyen espacios deben estar envueltos entre comillas. Por ejemplo, necesitas comillas para usar la fuente "Open Sans", pero no las necesitas para usar la fuente Lobster.

{Lección 8} Especifica cómo deben degradarse las fuentes:

Hay varias fuentes por defecto disponibles en todos los navegadores. Estas familias de fuentes genéricas incluyen a monospace, serif y sans-serif.

Cuando una fuente no está disponible en el sistema, puedes indicarle al navegador que "degrade" a otra fuente alternativa.

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

```
p {
  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.

{Lección 9} 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>
```

{Lección 10} Añadir bordes alrededor de tus elementos:

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

Por ejemplo, si quisiéramos crear un borde rojo de 5 píxeles alrededor de un elemento HTML, podríamos usar la siguiente clase

```
<style>
  .thin-red-border {
    border-color: red;
    border-width: 5px;
    border-style: solid;
  }
</style>
```

{Lección 11} Añadir esquinas redondeadas usando border-radius:

Podemos redondear esas esquinas con una propiedad CSS llamada border-radius (radio del borde).

Puedes especificar un *border-radius* usando píxeles como unidad de medida.

Dale a tu foto de gato un *border-radius* de *10px*.

Nota: Este desafío acepta múltiples soluciones posibles. Por ejemplo, puedes añadir border-radius a la clase .thick-green-border o a la clase .smaller-image.

{Lección 12} Hacer imágenes circulares usando border-radius (radio de borde):

Además de los píxeles, también puedes especificar el border-radius usando un porcentaje.

Dale a tu foto de gato un border-radius de 50%.

{Lección 13} Establecer el color de fondo de un elemento div:

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

Por ejemplo, si quieres que el color de fondo de un elemento sea green, lo harías dentro de tu elemento style

```
.green-background {
  background-color: green;
}
```

{Lección 14} Establecer el id de un elemento:

Además de las clases, cada elemento HTML también puede tener un atributo id.

Existen varios beneficios de usar atributos de id: Puedes usar un id para dar estilo a un elemento específico; más adelante aprenderás que puedes usar atributos "id" para seleccionar y modificar elementos específicos por medio de JavaScript.

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.

A continuación te mostramos un ejemplo de cómo asignar a tu elemento h2 el id cat-photo-app

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

{Lección 15} Usar un atributo de id para aplicar estilo a un elemento:

Una cosa genial de los atributos id es que, al igual que las clases, puedes aplicarles estilos con CSS.

Sin embargo, un id no es reutilizable y solo debe aplicarse a un único elemento. Un id también tiene mayor especificidad (importancia) que una clase, así que si aplicas un id y una clase al mismo elemento y sus estilos se contradicen, se aplicarán los estilos del id.

A continuación te mostramos un ejemplo de cómo puedes tomar tu elemento con el atributo id llamado cat-photo-element y asignarle el color de fondo verde. En tu elemento style agrega la siguiente declaración

```
#cat-photo-element {
  background-color: green;
}
```

Nota: Ten en cuenta que dentro de tu elemento style siempre debes hacer referencia a las clases agregándoles un punto . adelante del nombre. Para hacer referencia a un id, debes agregar # delante de su nombre.

{Lección 16} Ajustar el padding (relleno) de un elemento:

Dejaremos de lado por un momento nuestra Cat Photo App y aprenderemos un poco más sobre como aplicar estilos al HTML.

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.

{Lección 17} Ajustar el margen de un elemento:

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

Aquí podemos ver que la caja azul y la caja roja están anidadas dentro de la caja amarilla. Ten en cuenta que la caja roja tiene un margin más grande que el de la caja azul, lo que hace que aparezca más pequeña.

Cuando aumentas el margin de la caja azul, esto aumenta la distancia entre su borde y los elementos que la rodean.

{Lección 18} Añadir un margen negativo a un elemento:

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

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

{Lección 19} Añadir 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.

{Lección 20} Añadir 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.

{Lección 21} 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;

Estos cuatro valores se leen en el sentido de las agujas del reloj: arriba, derecha, abajo, izquierda, (top, right, bottom, left) y producirán exactamente el mismo resultado que usar las instrucciones específicas de padding.

{Lección 22} Utilizar 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;

Estos cuatro valores se leen en el sentido de las agujas del reloj: arriba, derecha, abajo, izquierda, (top, right, bottom, left) y producirán exactamente el mismo resultado que usar las instrucciones específicas de margen.

{Lección 23} Usar 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.

Volvamos a nuestra CatPhotoApp para practicar el uso de selectores CSS.

Para este desafío, usarás el selector de atributos *[attr=value]* para aplicar estilo a las casillas de verificación (checkboxes) en CatPhotoApp. Este selector busca estilos que tengan un valor de atributo específico. 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;
}
```

{Lección 24} Unidades absolutas y relativas 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.

- Ejemplo: Añadir una propiedad padding al elemento con clase red-box y establécelo en 1.5em.

{Lección 25} Aplicar un estilo al elemento HTML body:

Ahora vamos comenzar de cero y hablaremos sobre la herencia CSS (en inglés: "CSS inheritance").

Toda página HTML tiene un elemento body.

Para demostrar que el elemento body existe aquí, podemos asignarle un background-color black (negro).

Para ello, agregamos la siguiente declaración a nuestro elemento style

```
body {
  background-color: black;
}
```

{Lección 26} Heredar estilos del elemento body:

Ahora hemos demostrado que cada página HTML tiene un elemento body, y que a este elemento body también se le puede dar estilo con CSS.

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

Primero, crea un elemento h1 con el texto Hello World

Luego, demos el color green (verde) a todos los elementos de tu página, añadiendo color: green; a tu declaración de estilo del elemento body.

Finalmente, da a tu elemento body un valor para font-family de monospace añadiendo font-family: monospace; a la declaración de estilo del elemento body.

{Lección 27} Priorizar un estilo por sobre otro:

A veces los elementos HTML reciben múltiples estilos que entran en conflicto entre sí.

Por ejemplo, tu elemento h1 no puede ser verde y rosado al mismo tiempo.

Veamos qué ocurre cuando creamos una clase que hace que el texto sea rosado ("pink"), y luego se la aplicamos a un elemento. ¿Sobreescribirá nuestra clase la prioridad CSS color: green; del elemento body?

Respuesta: ¡Nuestra clase pink-text sobrescribió la declaración CSS de nuestro elemento body!

{Lección 28} Sobreescribir estilos en código CSS posterior:

¿qué podemos hacer para sobrescribir nuestra clase pink-text?

Crea una clase CSS adicional llamada blue-text que asigne a un elemento el color azul ("blue"). Asegúrate de que esté debajo de tu declaración de pink-text.

Aplica la clase blue-text a tu elemento h1 además de tu clase pink-text, y veamos cuál de las dos gana.

Para aplicar múltiples atributos de clase a un elemento HTML debes dejar un espacio entre ellos, como se muestra a continuación

class="class1 class2"

Nota: No importa el orden en que las clases estén enlistadas dentro del elemento HTML.

Sin embargo, lo importante es el orden de las declaraciones de class clases en la sección <style>. La segunda declaración siempre tendrá prioridad sobre la primera. Debido a que .blue-text ha sido declarada en segundo lugar, sobrescribirá los atributos de .pink-text

Acabamos de comprobar que los navegadores leen CSS desde arriba hacia abajo siguiendo el orden de las declaraciones. Por lo tanto, si llega a ocasionarse un conflicto, el navegador utilizará la última declaración CSS. Ten en cuenta que incluso si hubiésemos puesto blue-text antes que pink-text en nuestro elemento de clase h1, seguiría buscando al orden de la declaración y no al orden de su uso

{Lección 29} Sobreescribir declaraciones de clase dando estilo a atributos ID:

Hay otras formas de sobreescribir CSS. ¿Recuerdas los atributos id?

Sobrescribamos tus clases pink-text y blue-text, y haz que el h1 sea naranja al darle al elemento h1 un id y luego estilizando ese id.

Dale al elemento h1 el atributo id de orange-text.

<h1 id="orange-text">

Deja las clases blue-text y pink-text en tu elemento h1.

Crea una declaración CSS para el id orange-text en el elemento style.

```
#brown-text {
  color: brown;
}
```

Nota: No importa si declaras este CSS arriba o debajo de la clase pink-text, ya que el atributo id siempre tendrá precedencia

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.

{Lección 30} Sobreescribir declaraciones de clase con inline styles (estilos en línea):

Existen otras formas de sobreescribir código CSS.

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

<h1 style="color: green;">

Deja las clases blue-text y pink-text en tu elemento h1.

¡Excelente! Acabamos de demostrar que los estilos en línea sobrescribirán todas las declaraciones CSS en tu elemento style.

{Lección 31} Sobreescribir todos los demás estilos usando !important:

Pero, ¡aguarda! 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.

Volvamos a nuestra declaración de la clase pink-text. Recuerda que nuestra clase pink-text fue sobreescrita por declaraciones posteriores de clases, declaraciones de id e "inline styles".

Añade la palabra clave *!important* a la declaración de color de pink-text para asegurarte completamente de que tu elemento h1 será de color rosado.

Aquí te mostramos un ejemplo de cómo hacerlo

color: red !important;

{Lección 32} Utiliza código hexadecimal (hex code) para indicar colores específicos:

Sabías que hay otras maneras de representar colores en CSS? Una de estas formas se llama código hexadecimal, o código hex (hex code) para abreviar.

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.

{Lección 33} Utilizar código hexadecimal (hex code) para mezclar colores:

En código hexadecimal se utilizan 6 dígitos hexadecimales para representar los colores, dos para el componente rojo (R), verde (G) y azul (B).

¡A partir de estos tres colores puros (rojo, verde y azul) podemos variar las cantidades de cada componente de color para crear más de 16 millones de colores distintos!

Por ejemplo, el naranja es rojo puro mezclado con algo de verde y nada de azul. En hex code, esto se traduce como *#FFA500*.

El dígito 0 es el número más bajo en hex code, y representa la ausencia total de color.

El dígito F es el número más alto en hex code, y representa el brillo máximo.

{Lección 34} Usar código hexadecimal (hex code) abreviado:

Muchas personas se sienten abrumadas por tener más de 16 millones de colores posibles. Además, los códigos hexadecimales resultan difíciles de recordar. Afortunadamente, puedes abreviar gran parte de ellos.

Por ejemplo, el código hexadecimal *#FF0000* del color rojo puede acortarse a *#F00.* Esta forma abreviada utiliza un dígito para el rojo, un dígito para el verde, y un dígito para el azul.

Esto reduce el número total de colores posibles a alrededor de 4.000. Sin embargo, los navegadores interpretarán que *#FF0000* y *#F00* son exactamente el mismo color.

{Lección 35} Utilizar 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(0, 0, 0)

El valor RGB del color blanco se ve así

rgb(255, 255, 255)

En lugar de usar seis dígitos hexadecimales, como hacemos con el código hexadecimal, en RGB se especifica el brillo de cada color con un número que va de 0 a 255.

Si haces el cálculo, cada uno de los dos dígitos para un color representa 16 combinaciones, lo que nos da 256 valores posibles. Entonces, RGB, que comienza a contar desde cero, tiene el mismo número exacto de valores posibles que el código hexadecimal.

A continuación puedes ver un ejemplo de cómo cambiar el color de fondo de body a naranja usando su código RGB.

```
body {
  background-color: rgb(255, 165, 0);
}
```

{Lección 36} Utilizar RGB para mezclar colores:

Al igual que con el código hexadecimal, puedes mezclar colores combinando valores RGB.

{Lección 37} Usar 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.

Sigue las instrucciones a continuación para ver cómo puedes cambiar el estilo de varios elementos modificando únicamente tres valores.

- Ejercicio del freecode → En la clase penguin, cambia el valor black a gray, el valor gray a white y el valor yellow a orange.

{Lección 38} Crear una variable de CSS personalizada:

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;

Esto creará una variable llamada --penguin-skin y le asignará el valor gray. Ahora puedes usar esa variable en cualquier otro lugar de tu código CSS para cambiar el valor de otros elementos a "gray" (gris).

- Ejercicio del freecode→ En la clase penguin, crea un nombre de variable --penguin-skin y asígnale el valor gray.

{Lección 39} Utilizar una variable de CSS personalizada:

Luego de crear tu variable, puedes asignar su valor a otras propiedades CSS haciendo referencia a su nombre.

background: var(--penguin-skin);

Esto cambiará el fondo de cualquier elemento que utilice esta variable a "gray" (gris) porque ese es el valor de la variable --penguin-skin. Ten en cuenta que los estilos no se aplicarán a menos que los nombres de las variables utilizados estén escritos exactamente igual.

- ejercicio en el freecode --> Aplica la variable --penguin-skin a la propiedad background de las clases penguin-top, penguin-bottom, right-hand y left-hand.

{Lección 40} 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.

Nota: Este valor de respaldo no se utiliza para aumentar la compatibilidad con otras versiones de navegadores y no funcionará en navegadores IE (Internet Explorer). Más bien, se utiliza para que el navegador tenga un color para mostrar si no encuentra tu variable.

A continuación te mostramos como se hace

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

Esto establecerá el "background" (color de fondo) a black (negro) si tu variable no está establecida. Fíjate que esto puede ser útil para la depuración de errores.

{Lección 41} Mejorar la compatibilidad con navegadores por medio de configuraciones de respaldo o "browser fallbacks":

Cuando trabajes con código CSS, posiblemente te enfrentarás en algún momento con problemas de compatibilidad con otros navegadores web. Por esta razón es importante proporcionar configuraciones de respaldo para otros navegadores o "browser fallbacks" para lidiar con posibles problemas de compatiblidad.

Cuando tu navegador analiza el código CSS de una página web, ignora cualquier propiedad que no reconozca o soporte. Por ejemplo, si utilizas una variable CSS para asignar un color de fondo en un sitio, Internet Explorer ignorará el color de fondo establecido porque no soporta el uso de variables CSS. En ese caso, el navegador utilizará cualquier valor que tenga establecido como valor de esa propiedad. Si no puede encontrar ningún otro valor establecido para esa propiedad en el código, se revertirá al valor por defecto de ese navegador, lo que habitualmente no será lo ideal.

Esto significa que si quieres proporcionar un valor de respaldo para el navegador, esto es tan sencillo como incluir otro valor más comúnmente soportado inmediatamente antes de tu

declaración. De este modo, un navegador antiguo tendrá ún valor que sí pueda soportar, mientras que un navegador más nuevo interpretará cualquier declaración que incluyas más adelante en la cascada.

- ejercicio en el freecode--> Parece que en este código se utiliza una variable para establecer el color de fondo de la clase .red-box. Mejoremos la compatibilidad de nuestro código con otros navegadores, añadiendo otra declaración de background justo antes de la declaración existente, y estableciendo este valor de respaldo como red (rojo)

{Lección 42} Heredar variables CSS:

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.

- ejercicio en el freecode--> Define una variable llamada --penguin-belly en el selector :root y asígnale el valor pink (rosado). Luego podrás ver que la variable es heredada y que todos los elementos secundarios que la utilizan tendrán el color de fondo rosado.

{Lección 43} Cambiar una variable para un área específica:

Cuando creas tus variables en :root, el valor de esa variable quedará establecido para toda la página.

Luego podrás sobreescribir estas variables, configurándolas de nuevo dentro de un elemento específico.

- ejercicio en el freecode-->Cambia el valor de --penguin-belly a white en la clase penguin.

{Lección 44} Usar un media query para cambiar una variable:

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.

- ejercicio en el freecode--> Modifica el selector :root de la media query para que --penguin-size sea redefinido y reciba un valor de 200px. Además, redefine --penguin-skin y asígnale el valor black (negro). Luego, cambia el tamaño de la vista previa para ver este cambio en acción.

[Diseño Visual Aplicado]

{Lección 1} Crear un balance visual usando la propiedad *text-align*:

Esta sección del currículo se enfoca en el Diseño Visual Aplicado. El primer grupo de desafíos se basa en el diseño de la tarjeta provista para mostrar un número de principios fundamentales.

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

text-align: justify; hace que todas las líneas de texto, excepto la última línea, se encuentren con los lados izquierdo y derecho de la caja.

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.

{Lección 2} Ajustar el ancho de un elemento utilizando la propiedad *width*:

Puedes especificar el ancho de un elemento con la propiedad width en CSS. Los valores se pueden dar en unidades de longitud relativa (como em, unidades de longitud absoluta (como px, o como un porcentaje de su elemento padre contenedor. El siguiente ejemplo cambia el ancho de una imagen a 220px

```
img {
  width: 220px;
}
```

{Lección 3} Ajustar la altura de un elemento utilizando la propiedad *height*:

Puedes especificar la altura de un elemento con la propiedad height en CSS como lo haces con la propiedad width (ancho). En el siguiente ejemplo, se cambia la altura de una imagen a 20px:

```
img {
  height: 20px;
}
```

{Lección 4} Utilizar la etiqueta *strong* para poner el texto en negrita:

Para poner el texto en negrita, puedes usar la etiqueta *strong*. Esto se usa a menudo para llamar la atención sobre el texto y simbolizar que es importante. Con la etiqueta *strong*, el navegador aplica el CSS de *font-weight: bold;* al elemento.

- ejercicio en el freecode-->>Envuelve una etiqueta *strong* alrededor del texto de Stanford University dentro de la etiqueta p (no incluyas el punto).

{Lección 5} Utilizar la etiqueta *u* para subrayar texto:

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.

Nota: Trata de evitar el uso de la etiqueta u, puesto que podría confundirse con un enlace. Las etiquetas de enlaces tienen un formato subrayado por defecto.

- ejercicio del freecode--> Envuelve la etiqueta u solo alrededor del texto Ph.D. students

{Lección 6} Usar la etiqueta *em* para poner texto en cursiva:

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.

- Ejercicio de Freecode→ Envuelve una etiqueta em alrededor del contenido de la etiqueta de párrafo para darle énfasis.

{Lección 7} Usar la etiqueta *s* para tachar texto:

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.

- ejercicio del freecode-->Envuelve la etiqueta s alrededor de Google dentro de la etiqueta h4

{Lección 8} Crear una línea horizontal usando el elemento *hr*:

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.

Nota: en HTML, la etiqueta hr se cierra sola, por lo tanto no necesita una etiqueta de cierre por separado.

- ejercicio del freecode-->Agrega una etiqueta hr debajo de h4 que contiene el título de la tarjeta.

{Lección 9} Ajustar 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

Los valores RGB pueden variar de 0 a 255. El valor alfa puede variar de 1, que es completamente opaco o un color sólido, a 0, que es completamente transparente o claro. rgba() es ideal para usar en este caso, ya que te permite ajustar la opacidad. Esto significa que no tienes que bloquear completamente el fondo.

- ejercicio del freecode-->Utilizarás background-color: rgba(45, 45, 45, 0.1) para este desafío. Produce un color gris oscuro que es casi transparente dado el bajo valor de opacidad de 0.1.
  Para que el texto destaque más, ajusta el background-color del elemento h4 al valor rgba() dado.
  También para el h4, elimina la propiedad height y agrega padding de 10px.

{Lección 10} Ajustar el tamaño de un título contra una etiqueta de párrafo:

El tamaño de fuente de las etiquetas de encabezado (h1 a h6) generalmente debería ser mayor que el tamaño de fuente de las etiquetas de párrafos. Esto hace que sea más sencillo para que el usuario entienda visualmente el diseño y el nivel de importancia de cada elemento en la página. Utiliza la propiedad *font-size* para ajustar el tamaño del texto en un elemento.

- ejercicio del freecode-->Para que el encabezado sea significativamente más grande que el párrafo, cambia la etiqueta font-size de la etiqueta h4 a 27 píxeles.

{Lección 11} Agregar *box-shadow* a un elemento tipo tarjeta

La propiedad *box-shadow* aplica una o más sombras a un elemento.

La propiedad *box-shadow* toma valores para

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 y

color, en ese orden.

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.

A continuación un ejemplo de CSS para crear múltiples sombras con un poco de desenfoque, con colores negros casi transparentes

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

- ejercicio del freecode-->El elemento ahora tiene un id de thumbnail. Con este selector, usa el los valores del ejemplo CSS anterior para aplicar box-shadow sobre la tarjeta.

{Lección 12} 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.

El valor dado se aplicará a todo el elemento, ya sea una imagen con cierta transparencia, o los colores de primer plano y fondo para un bloque de texto.

- ejercicio del freecode-->Establece la opacity de las etiquetas de anclajes en 0.7 usando la clase links para seleccionarlas.

{Lección 13} Usar 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*"

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

- ejercicio del freecode-->Transforma el texto de h4 en mayúsculas usando la propiedad text-transform.

{Lección 14} Establecer el tamaño de fuente para varios elementos de títulos:

La propiedad *font-size* se usa para especificar que tan grande es el texto en un elemento dado. Esta regla se puede utilizar para varios elementos para crear coherencia visual del texto en una página. En este desafío, establecerá los valores para todas las etiquetas h1 a h6 para equilibrar los tamaños de los títulos.

- ejercicio del freecode-->En las etiquetas style, establece el font-size de
  Etiqueta h1 a 68px.
  Etiqueta h2 a 52px.
  Etiqueta h3 a 40px.
  Etiqueta h4 a 32px.
  Etiqueta h5 a 21px.
  Etiqueta h6 a 14px.

{Lección 15} Establecer el *font-weight* para varios elementos de títulos:

Se establece el font-size de cada etiqueta de título en el último desafío, aquí podrás ajustar el *font-weight*.

La propiedad *font-weight* establece que tan gruesos o delgados son los caracteres en una sección de texto.

- ejercicio del freecode-->
  Establece el font-weight de la etiqueta h1 en 800.
  Establece el font-weight de la etiqueta h2 en 600.
  Establece el font-weight de la etiqueta h3 en 500

{Lección 16} Establecer el tamaño de fuente del texto del párrafo:

La propiedad *font-size* en CSS no se limita a los títulos, se puede aplicar a cualquier elemento que contenga texto.

- ejercicio del freecode--> Cambia el valor de la propiedad *font-size* para el párrafo a 16px para hacerlo más visible.

{Lección 17} 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.

- ejercicio del freecode-->Agrega una propiedad line-height a la etiqueta p y establezca en 25px.

{Lección 18} Ajustar el *hover* de una etiqueta anchor:

Este desafío mostrará el uso de las *pseudo-clases*. 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

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

- ejercicio en el freecode-->El editor de código tiene una regla CSS para dar estilo a todas las etiquetas a negras. Añade una regla para que cuando el usuario pase sobre la etiqueta a, el color `` sea azul.

{Lección 19} Cambiar la posición relativa de un elemento:

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;
}
```

Cambiando la posición de un elemento a *relative* no lo quita del flujo normal; *otros elementos a su alrededor aún se comportarán como si dicho elemento estuviera en su posición predeterminada.*

Nota: el posicionamiento te da mucha flexibilidad y poder sobre el diseño visual de una página. Es bueno recordar que no importa la posición de los elementos, el lenguaje de marcado HTML subyacente debe organizarse y tener sentido cuando se lee de arriba a abajo. Así es como los usuarios con discapacidades visuales (que dependen de dispositivos de asistencia como lectores de pantalla) acceden a tu contenido.

- ejercicio del freecode-->Cambia la propiedad position de h2 a relative, y usa CSS desplazamiento para moverlo 15 pixeles lejos de la parte superior top donde se ubica dentro del flujo normal. Observa que no hay impacto en las posiciones de los elementos h1 y p que están al rededor.

{Lección 20} Mover un elemento posicionado relativamente con desplazamientos de CSS:

Los desplazamientos CSS de top o bottom y left o right indican al navegador hasta que punto debe compensar un elemento en relación con el lugar donde se ubicara en el flujo normal del documento. Está compensando un elemento lejos de un punto dado, lo que aleja el elemento del lado al que se hace referencia (efectivamente, en la dirección opuesta). Como viste en el

último desafío, usando el desplazamiento top movió el h2 hacia abajo. Del mismo modo, usando un desplazamiento left mueve un elemento hacia la derecha.

- ejercicio del freecode-->Utiliza los desplazamientos CSS para mover los h2 15 píxeles a la derecha y 10 píxeles hacia arriba.
  Respuesta el ejercicio-->

  h2 {

  position: relative;
  left:15px;
  bottom:10px;

  }

{Lección 21} Bloquear 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.

- ejercicio del freecode-->Bloquea el elemento #searchbar en al parte superior derecha de su section padre declarando su positioncomo absolute. Dale desplazamientos top y right de 50 píxeles cada uno.

{Lección 22} Bloquear 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.

- ejercicio del freecode-->La barra de navegación en el código está etiquetada con un id de navbar. Cambia su position para fixed, y el desplazamiento 0 píxeles del top y 0 píxeles de la left. Después de agregar el código, desplázate por la ventana de vista previa para ver como la navegación permanece en su lugar.

{Lección 23} Empujar 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.

- ejercicio del freecode-->El lenguaje de marcado dado funcionaria bien como un diseño de dos columnas, con los elementos section y aside uno al lado del otro. Da el #left elemento float de left y #right elemento float de right.

{Lección 24} Cambiar 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.

- ejercicio del freecode-->Agrega una propiedad z-index al elemento con la clase first (el rectángulo rojo) y asígnale un valor de 2 para que cubra al otro elemento (rectángulo azul).

{Lección 25} Centrar 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.

- ejercicio del freecode-->Centra el div en la página agregando una propiedad margin con un valor de auto.

{Lección 26} Aprende sobre colores complementarios:

La teoría del color y su impacto en el diseño es un tema pesado y solo cubriremos los aspectos básicos en los próximos desafíos. En un sitio web, los colores llaman la atención, provocan emociones y crean una armonía visual. Con diferentes combinaciones de colores se puede cambiar el aspecto de un sitio web y requiere una planificación extensa decidirse por una paleta de color que se integre con nuestro contenido.

El círculo cromático es una herramienta útil para observar cómo los colores están relacionados entre sí - es un círculo donde los tonos similares están juntos y los tonos diferentes alejados. Cuando dos colores opuestos están juntos en el círculo, se los llama colores complementarios. Se caracterizan porque si se combinan, se cancelan así mismos y crean un color gris. Sin embargo, al ubicarse juntos, estos colores parecen más brillantes y producen un contraste visual fuerte.

A continuación hay algunos ejemplos de colores con sus códigos hexadecimales

rojo (#FF0000) y cian (#00FFFF)

verde(#00FF00) y magenta (#FF00FF)

azul (#0000FF) y amarillo (#FFFF00)

Esto es diferente del anticuado modelo de color RYB que muchos de nosotros aprendimos en la escuela, que tiene diferentes colores primarios y complementarios. La teoría moderna del color utiliza el modelo aditivo RGB (como en una pantalla de computadora) y el modelo restante CMY(K) (como en la impresión). Lee aquí para obtener más información sobre este complejo tema.

Hay muchas herramientas de selección de color disponibles en línea que tienen la opción de encontrar el complemento de un color.

Nota: El uso del color puede ser una forma poderosa de agregar interés visual a una página. Sin embargo, el color por sí solo no debe utilizarse como la única manera de transmitir información importante porque los usuarios con deficiencias visuales pueden no entender ese contenido. Esta cuestión se tratará con más detalle en los desafíos de accesibilidad aplicada.

- ejercicio del freecode -->Cambia la propiedad background-color de las clases blue y yellow a sus colores respectivos. Observa cómo se ven los colores unos a otros distintos de los que se comparan con el fondo blanco.

{Lección 27} Aprende sobre colores terciarios:

Los monitores y las pantallas crean diferentes colores al combinar cantidades de luz roja, verde y azul. Esto se conoce como modelo aditivo de color RGB en la teoría de moderna de color. Rojo (R), verde (G) y azul (B) «por sus siglas en inglés» son colores primarios. Al combinar dos colores primarios se los colores secundarios cian (G + B), magenta (R + B) y amarillo (R + G). Ya viste estos colores en los desafíos de colores complementarios. Estos colores secundarios son el complemento del color primario no utilizado en su creación y están frente a ese color primario en el círculo cromático. Por ejemplo, el magenta está compuesto de rojo y azul y es el complemento del verde.

Los colores terciarios se forman al combinar dos colores primarios con uno de sus vecinos de color secundario. Por ejemplo, entre el modelo de color RGB, rojo (primario) y amarillo (secundario) forman naranja (terciario). Esto añade seis colores a un círculo cromático simple para un total de doce.

Hay varios métodos para seleccionar colores diferentes que resultan de una combinación armoniosa en diseño. Un ejemplo que puede usar colores terciarios es el esquema de color complementario dividido. Este esquema comienza con un color base, luego lo empareja con los dos colores que están adyacentes a su complemento. Los tres colores proporcionan un fuerte contraste visual en un diseño, pero son más sutiles que utilizar dos colores complementarios.

Aquí hay tres colores creados usando el esquema de dividir-complemento

Color Código hexadecimal

anaranjado *#FF7F00*

cian *#00FFFF*

frambuesa *#FF007F*

- ejercicio del freecode-->Cambia la propiedad background-color de las clases orange, cyan y raspberry a sus colores respectivos. Asegúrate de usar los códigos hexadecimales y no los nombres de colores.

{Lección 28} Ajustar los colores de varios elementos para colores complementarios:

En el desafío de colores complementarios vimos que al colocar dos colores opuestos del círculo cromático, parecen más vivos. Sin embargo, el contraste visual fuerte puede ser molesto si se utiliza en un sitio web y algunos veces pueden hacer que el texto sea difícil de leer si está dentro de un complementary-color background. En la práctica, usualmente se usa uno de los colores como dominante y los complementarios se usan para atraer atención visual a cierto contenido dentro de la página.

- ejercicio del freecode-->Está página utilizará una sombra de cerceta (#09A7A1) como color dominante y el complementario naranja (#FF790E) para resaltar los botones de inicio de sesión. Cambia el background-color del header y footer de negro a cerceta. Después, cambia h2 texto color también a cerceta. Por último, pon naranja el background-color del button.

{Lección 29} Ajustar el matiz de un color:

Los colores tienen varias características tales como el matiz, la saturación y la ligereza. CSS3 introdujo la propiedad hsl() como una forma alternativa de elegir un color indicando directamente estas características.

Se suele pensar que Hue es el "color". Si imaginas un espectro de colores con un rojo en la izquierda que se torna verde en el medio y azul en la derecha, el tono es donde cabe un color a lo largo de esta línea. En hsl(), el tono usa un concepto de círculo cromático en lugar del espectro, donde el ángulo del color en el círculo se da como un valor entre 0 y 360.

Saturation es la cantidad de gris en un color. Un color totalmente saturado no tiene gris y un color mínimamente saturado es casi completamente gris. Esto se da como un porcentaje con 100% de saturación.

Lightness es la cantidad de blanco en un color. Un porcentaje se da desde 0% (negro) hasta 100% (blanco), donde 50% es el color normal.

Aquí hay algunos ejemplos de hsl() con colores de iluminación normales y completamente saturados

Color HSL

rojo *hsl(0, 100 %, 50 %)*

amarillo *hsl(60, 100 %, 50 %)*

verde *hsl(120, 100 %, 50 %)*

cian *hsl(180, 100 %, 50 %)*

azul *hsl(240, 100 %, 50 %)*

magenta *hsl(300, 100 %, 50 %)*

- ejercicio del freecode--> Cambia el background-color de cada elemento div sobre la base de los nombres de las clases (green, cyan o blue) utilizando hsl(). Los tres deben tener una saturación completa y una iluminación normal.

{Lección 30} Ajustar el tono de un color:

La opción hsl() en CSS también hace que sencillo ajustar el tono de un color. Mezclar blanco con un tono puro crea un tinte de ese color y añadir negro hará una sombra. De forma alternativa, un tono se produce al añadir gris o tintes y sombras. Recuerda que la 's' y 'l' del hsl() representan saturación y ligereza, respectivamente. El porcentaje de saturación cambia la cantidad de gris y el porcentaje de luz determina el porcentaje de blanco o de negro que hay en el color. Esto es útil cuando se tiene un tono base que se quiere, pero necesita variaciones diferentes del mismo.

- ejercicio del freecode-->Todos los elementos tienen un background-color predeterminado de transparent. Nuestro elemento nav parece tener un fondo cyan ya que el elemento detrás del mismo tiene un background-color establecido a cyan. Añade un background-color al elemento nav para que use el mismo tono de cyan, pero que tenga 80% de saturación y luminosidad 25% para cambiar su tono y sombra.

{Lección 31} Crear 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, ...);

El primer argumento especifica la dirección desde la que comienza la transición de color, se puede establecer como un grado, donde 90deg hace un gradiente horizontal (de izquierda a derecha) y 45deg hace un gradiente diagonal (de abajo a izquierda hacia arriba a la derecha). Los siguientes argumentos especifican el orden de los colores utilizados en el degradado. Ejemplo :

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

- ejercicio del freecode-->Utiliza un linear-gradient() para el background del elemento div y configúralo desde una dirección de 35 grados para cambiar el color de #CCFFFF a #FFCCCC.

{Lección 32} Utilizar un degradado lineal CSS para crear un elemento rayado:

La función repeating-linear-gradient() es muy similar a linear-gradient() con la principal diferencia de que repite el patrón de degradado especificado. repeating-linear-gradient() acepta una variedad de valores, pero para simplificar, trabajarás con un valor de ángulo y valores de parada de color en este desafío.

El valor del ángulo es la dirección del gradiente. Las paradas de color son como valores de ancho que marcan donde tiene lugar una transición, y se dan con un porcentaje o un número de píxeles.

En el ejemplo demostrado en el editor de código, el degradado comienza con el color yellow a 0 píxeles que se funde en el segundo color blue a 40 píxeles de distancia desde el principio. Puesto que la siguiente parada de color también es de 40 píxeles, el degradado cambia inmediatamente al tercer color green, el cual se funde en el cuarto valor de color red ya que está a 80 píxeles del principio del degradado.

Para este ejemplo, ayuda a pensar en las paradas de color como pares donde cada dos colores se mezclan juntos.

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

Si cada dos valores de parada de color son del mismo color, la mezcla no es visible porque está entre el mismo color, seguido de una dura transición hacia el siguiente color, así que terminas con rayas.

- ejercicio del freecode -->Haz rayas cambiando el repeating-linear-gradient() para usar un ángulo gradiente de 45deg, luego ajusta las dos primeras paradas de color en yellow, y finalmente las dos segundas paradas de color en black.

{Lección 33} Crear 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.

- ejercicio del freecode-->Usando el enlace https://cdn-media-1.freecodecamp.org/imgr/MJAkxbh.png, asigna el fondo background de toda la página con el selector body.

{Lección 34} Utilizar 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);
```

}

- ejercicio del freecode-->Aumenta el tamaño del elemento con el id de ball2 a 1.5 veces su tamaño original.

{Lección 35} Utilizar la propiedad de escala de transformación CSS para escalar un elemento al desplazarse (pasar el mouse):

La propiedad transform tiene una variedad de funciones que el 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.

Aquí hay un ejemplo para escalar los elementos de párrafo a 2.1 veces su tamaño original

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

Nota: La aplicación de una transformación a un elemento div también afectará a cualquier elemento secundario contenido del div.

- ejercicio del freecode-->Agrega una regla CSS para el estado hover del div y usa la propiedad transform para escalar el elemento div a 1.1 veces su tamaño original cuando un usuario pasa sobre él.

{Lección 36} Utilizar la propiedad de transformación CSS skewX para inclinar un elemento a lo largo del eje X:

La siguiente función de la propiedad transform es skewX(), que inclinar el elemento seleccionado a lo largo de su eje X (horizontal).

El siguiente código inclina el elemento de párrafo en -32 grados a lo largo del eje X.

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

- ejercicio del freecode-->Inclina el elemento con el id de bottom en 24 grados a lo largo del eje X utilizando la propiedad transform.

{Lección 37} Utilizar la propiedad de transformación CSS skewY para inclinar un elemento a lo largo del eje Y:

Dado que la función skewX() inclina el elemento seleccionado a lo largo del eje X en un grado dado, no sorprende que la propiedad skewY() incline un elemento a lo largo del eje Y (vertical).

- ejercicio del freecode-->Inclina el elemento con el id de top -10 grados a lo largo del eje Y utilizando la propiedad transform.

{Lección 38} Crear un gráfico usando CSS:

Al manipular diferentes selectores y propiedades, puedes hacer figuras interesantes. Una de las figuras más fáciles de intentar es la luna creciente. Para este desafío necesitas trabajar con la propiedad box-shadow que aplica la sombra de un elemento, junto con la propiedad border-radius que controla la redondez de las esquinas del elemento.

Crearás un objeto redondo y transparente con una sombra nítida que está ligeramente desplazada hacia un lado - la sombra en realidad va a ser la figura de luna que verás.

Para crear un objeto redondo, la propiedad border-radius se le debe asignar un valor de 50%.

Puede que recuerdes de un desafío anterior que la propiedad box-shadow toma valores para offset-x, offset-y, blur-radius, spread-radius y un valor para el color, en ese orden. Los valores blur-radius y spread-radius son opcionales.

- ejercicio del freecode-->Manipula el elemento cuadrado en el editor para crear la figura de luna. Primero, cambia el background-color a transparent, luego establece la propiedad border-radius en 50% para hacer la forma circular. Finalmente, cambia la propiedad box-shadow para asignar offset-x a 25px, offset-y a 10px, blur-radius a 0, spread-radius a 0 y el color a blue.

{Lección 39} Como funcionan las propiedades de CSS @keyframes y animación:

Para animar un elemento, necesitas conocer las propiedades de animación y la regla @keyframes. Las pro piedades 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%.

- ejercicio del freecode-->Crea una animación para el elemento con el id rect, estableciendo animation-name en rainbow y animation-duration a 4 segundos. A continuación, declara una regla @keyframes y estableca el background-color al principio de la animación (0%) en azul, el centro de la animación (50%) en verde y el final de la animación (100%) en amarillo.

{Lección 40} Usar animación CSS para cambiar el estado del desplazamiento de un botón (Hover):

Puedes usar CSS @keyframes para cambiar el color de un botón en su estado de desplazamiento.

Aquí hay un ejemplo de como cambiar el ancho de una imagen al pasar sobre ella

```
<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" />
```

- ejercicio del freecode-->Ten en cuenta que ms significa milisegundos, donde 1000ms es igual a 1s.
  Utiliza CSS @keyframes para cambiar el background-color del elemento button para que se convierta en #4791d0 cuando un usuario pase sobre él. La regla @keyframes solo debe tener una entrada para 100%.

{Lección 41} Modificar el modo de relleno de una animación (animation-fill-mode):

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;
```

- ejercicio del freecode-->Establece la propiedad animation-fill-mode de button:hover en forwards para que el botón permanezca resaltado cuando un usuario pase sobre el.

{Lección 42} 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%).

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

- ejercicio del freecode-->Agrega un movimiento horizontal a la animación div. Usando la propiedad desplazamiento left, agrega a la regla @keyframes para que el arcoíris (rainbow) comience en 0 píxeles en 0%, se mueva a 25 píxeles en 50%, y termine en -

25 píxeles en 100%. No reemplaces la propiedad top en el editor; la animación debe tener movimiento vertical y horizontal.

{Lección 43} 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.

- ejercicio del freecode-->Apunta al elemento con el id de ball y agrega la propiedad opacity establecida en 0.1 en 50%, de modo que el elemento se desvanezca a medida que se mueve hacia la derecha.

{Lección 44} 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. Por ejemplo

animation-iteration-count: 3;

En este caso, la animación se detendrá después de ejecutarse 3 veces, pero es posible hacer que la animación se ejecute continuamente estableciendo ese valor en *infinite*.

- ejercicio del freecode-->Para mantener la bola rebotando a la derecha en un bucle continuo, cambia la propiedad animation-iteration-count a infinite.

{Lección 45} Haz latir un corazón con CSS usando un recuento de animación infinita:

Aquí hay un ejemplo más de animación con la propiedad animation-iteration-count que usa el corazón que diseñaste en un desafío anterior.

La animación del latido de un segundo consta de dos piezas animadas. Los elementos heart (incluyendo las piezas :before y :after) se animan para cambiar el tamaño usando la propiedad transform, y el fondo div se anima para cambiar su color usando la propiedad background.

- ejercicio del freecode-->Mantén el corazón latiendo agregando la propiedad animation-iteration-count tanto para la clase back como para la clase heart y estableciendo el valor en infinite. Los selectores heart:before y heart:after no necesitan ninguna propiedad de animación.

{Lección 46} Elementos animados con fluctuaciones:

Hay muchas formas de alterar la cantidad de animaciones de elementos similares con animaciones. Hasta ahora, esto se ha logrado al aplicar una *propiedad animation-iteration-count* y estableciendo reglas @keyframes.

A modo de ilustración, la animación de la derecha consta de dos estrellas, cada una de las cuales disminuye en tamaño y opacidad en la marca del 20% en la regla @keyframes, que crea la animación centelleante. Puede cambiar la regla @keyframes para uno de los elementos, así las estrellas titilan con diferentes ritmos.

- ejercicio del freecode-->Cambia la velocidad de animación del elemento con el nombre de la clase star-1 cambiando su regla @keyframes al 50%.

{Lección 47} Animar múltiples elementos con ritmos diferentes:

En el desafío anterior, cambiaste el ritmo de la animación para dos elementos similares animados alterando sus reglas @keyframes. Puedes lograr el mismo objetivo manipulando la animation-duration de múltiples elementos.

En la animación que se ejecuta en el editor de código, hay tres estrellas en el cielo que brillan a la misma velocidad en un ciclo continuo. Para hacerlos titilar con diferentes ritmos, puedes establecer la propiedad animation-duration con diferentes valores para cada elemento.

- ejercicio del freecode-->Establece la animation-duration de los elementos con las clases star-1, star-2y star-3 a 1s, 0.9s y 1.1s, respectivamente.

{Lección 48} Cambia la duración de las animaciones con palabras clave:

En las animaciones CSS, la propiedad *animation-timing-function* controla qué tan rápido un elemento animado cambia sobre la duración total de la animación. Si la animación es un carro moviéndose de un punto A a un punto B en un tiempo establecido (tu *animation-duration*), la *animation-timing-function* dicta cómo el carro acelera y desacelera durante el transcurso en marcha.

Existe un número de palabras clave predefinidas disponibles para las opciones populares. Por ejemplo, el valor por defecto es ease, que empieza despacio, acelera en el medio y luego reduce la velocidad de nuevo al final. Otras opciones incluye ease-out, que es rápida al inicio y luego reduce la velocidad, ease-in, que es lenta al inicio y luego acelera al final, o linear, que aplica una velocidad constante a lo largo de la animación.

- ejercicio del freecode-->Para los elementos con id ball1 y ball2, agrega una propiedad animation-timing-function a cada uno y asigna linear a #ball1 y ease-out a #ball2. Nota la diferencia entre cómo los elementos se mueven durante la animación pero terminan juntos, dado que comparten la misma animation-duration de 2 segundos.

{Lección 49} Como funcionan las curvas de Bezier:

El último desafío introdujo la propiedad animation-timing-function y algunas palabras clave que cambian la velocidad de una animación a lo largo de su duración. CSS ofrece una opción distinta a las palabras clave que proporciona un control aún más fino sobre cómo se desarrolla la animación, a través del uso de curvas Bezier.

En las animaciones CSS, las curvas Bezier se utilizan con la función cubic-bezier. La forma de la curva representa cómo se desarrolla la animación. La curva vive en un sistema de coordenadas de 1 a 1. El eje X de este sistema de coordenadas es la duración de la animación (piénsa en una escala de tiempo), y el eje Y es el cambio en la animación.

La función cubic-bezier consta de cuatro puntos principales que se encuentran en esta cuadrícula de 1 por 1: p0, p1, p2 y p3. p0 y p3 están configurados para ti: son los puntos inicial y final que siempre se encuentran respectivamente en el origen (0, 0) y (1, 1). Establece los valores x e y para los otros dos puntos, y donde los coloca en la cuadrícula determina la forma de la curva para la animación que debe seguir. Esto se hace en CSS declarando los valores x e y de los puntos "anchor" p1 y p2 de la siguiente forma:(x1, y1, x2, y2). Juntando todo, aquí hay un ejemplo de una curva de Bezier en codigo CSS

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

En el ejemplo anterior, los valores x e y son equivalentes para cada punto (x1 = 0.25 = y1 y x2 = 0.75 = y2), que si recuerdas la clase de geometría, da como resultado una línea que se extiende desde el origen hasta el punto (1, 1). Esta animación es un cambio lineal de un elemento durante la duración de una animación, y es lo mismo que usar la palabra clave linear. En otras palabras, cambia a una velocidad constante.

- ejercicio del freecode-->Para el elemento con el id de ball1, cambia el valor de la propiedad animation-timing-function de linear a su valor de función cubic-bezier equivalente. Utiliza los valores de puntos dados en el ejemplo anterior.

{Lección 50} Usar una curva de Bezier para mover un gráfico:

Un desafío anterior discutió la palabra clave ease-out que describe un cambio de animación que se acelera primero y luego se ralentiza al final de la animación. A la derecha, se muestra la diferencia entre la palabra clave ease-out (para el elemento azul) y la palabra clave linear (para el elemento rojo). Se pueden lograr progresiones de animación similares a la palabra clave ease-out utilizando una función de curva Bezier cúbica personalizada.

En general, el cambio de los puntos de anclaje p1 y p2 impulsa la creación de diferentes curvas Bezier, que controlan como la animación progresa a través del tiempo. Aquí hay un ejemplo de una curva de Bezier que usa valores para imitar el estilo de facilidad

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

Recuerda que todas las funciones cubic-bezier comienzan con p0 en (0, 0) y terminan con p3 en (1, 1). En este ejemplo, la curva se mueve más rápido a través del eje Y (comienza en 0, va a p1 y valor de 0, luego va a p2 y valor de 1) que se mueve a través del eje X (0 para iniciar, luego 0 para p1, hasta 0.58 para p2. Como resultado, el cambio en el elemento animado progresa más rápido que el tiempo de la animación para ese segmento. Hacia el final de la curva, la relación entre el cambio en los valores X e Y se invierte: el valor y se mueve de 1 a 1 (sin cambios), y los valores X se mueven de 0.58 a 1, lo que hace que los cambios de animación progresen más lentamente en comparación con la duración de la animación.

- ejercicio del freecode-->Para ver el efecto de esta curva Bezier en acción, cambia la animation-timing-function del elemento con id de red a una función cubic-bezier con valores x1, y1, x2, y2 establecidos respectivamente en 0, 0, 0.58, 1. Esto hará que ambos elementos progresen a través de la animación de manera similar.

{Lección 51} Hacer que el movimiento sea más natural usando una curva de Bezier:

Este desafío anima un elemento para replicar el movimiento de una pelota que se hace rebotes. Los desafíos anteriores cubrirán las curvas cúbicas de Bezier linear y ease-out, sin embargo, ninguna representa el movimiento de rebotes con precisión. Necesitas personalizar una curva de Bezier para esto.

La función animation-timing-function se realiza automáticamente en cada fotograma clave (keyframe) cuando el animation-iteration-count se establece en infinito. Dado que hay una regla de fotogramas clave establecida en el medio de la duración de la animación (en 50%, da como resultado dos progresiones de animación idénticas en el movimiento hacia arriba y hacia abajo de la pelota.

La siguiente curva cúbica de Bezier simula el movimiento de rebotes

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

Observa que el valor de y2 es mayor que 1. Aunque la curva cúbica de Bezier se mapea en un sistema de coordenadas 1 por 1, y solo puede aceptar valores x de 0 a 1, el valor y se puede

establecer en números mayores que uno. Esto da como resultado un movimiento de rebote que es ideal para simular la pelota rebotando.

- ejercicio del freecode-->Cambia el valor de la función animation-timing-function del elemento con el id de green a una función cubic-bezier con valores x1, y1, x2, y2 establecidos respectivamente en 0.311, 0.441, 0.444, 1.649.

[Accesibilidad Aplicada]

{Lección 1} Agregar un texto alternativo a las imágenes para accesibilidad de usuarios con dificultades de la vista:

Probablemente hayas visto un atributo *alt* en una etiqueta img en otros desafíos. El atributo alt describe el contenido de la imagen y proporciona un texto alternativo. Un atributo alt ayuda en los casos en que la imagen no se carga o un usuario no pueda verla. Los motores de búsqueda también lo utilizan para comprender qué contiene una imagen para incluirla en los resultados de búsqueda. Aquí hay un ejemplo

```
<img src="importantLogo.jpeg" alt="Company logo">
```

Las personas con dificultades visuales dependen de lectores de pantalla para convertir el contenido web a una interfaz de audio. Por esta razón, no podrán recibir la información si solo se les presenta de manera visual. En el caso de las imágenes, los lectores de pantalla pueden acceder el atributo alt y leer su contenido para proporcionar información clave.

Un buen texto alt le brinda al lector una breve descripción de la imagen. Siempre deberías incluir el atributo alt en tus imágenes. De acuerdo con las especificaciones de HTML5, esto ahora se considera obligatorio.

{Lección 2} Aprende cuando el texto alternativo debe dejarse en blanco:

En el último desafío, aprendiste que es obligatorio incluir un atributo alt al usar etiquetas img. Sin embargo, a veces las imágenes se agrupan con un título que ya las describe, o se usan solo para decoración. En estos casos el texto alt puede parecer redundante o innecesario.

Cuando una imagen ya se explica con el contenido de texto o no agrega significado a una página, img todavía necesita un atributo alt, pero se puede establecer en una cadena vacía. Aquí hay un ejemplo

```
<img src="visualDecoration.jpeg" alt="">
```

Las imágenes de fondo generalmente también caen bajo la etiqueta "decorativa". Sin embargo, normalmente se aplican con reglas CSS y, por lo tanto, no forman parte del proceso de lectores de pantalla del lenguaje de marcado.

Nota: Para imágenes con un título, es posible que aún desees incluir texto alt ya que ayuda a los motores de búsqueda a catalogar el contenido de la imagen.

{Lección 3} Usa títulos para mostrar relaciones jerárquicas de contenido:

Los títulos (h1 a h6 elementos) son etiquetas de caballo de batalla que ayudan a proporcionar estructura y etiquetado a su contenido. Los lectores de pantalla se pueden configurar para leer solo los títulos de una página para que el usuario obtenga un resumen. Esto significa que es importante que las etiquetas de los títulos en tu lenguaje de marcado tengan un significado semántico y se relacionen entre sí, no se elijan simplemente por sus valores de tamaño.

Significado semántico* significa que la etiqueta que usas alrededor del contenido indica el tipo de información que contiene.

Si estuvieras escribiendo un documento con una introducción, un cuerpo y una conclusión, no tendría mucho sentido poner la conclusión como una subsección del cuerpo en tu esquema. Debería ser su propia sección. Del mismo modo, las etiquetas de los títulos en una página web deben ir en orden e indicar las relaciones jerárquicas de su contenido.

Los títulos con rango igual (o superior) comienzan nuevas secciones implícitas, los títulos con rango inferior comienzan subsecciones de la anterior.

Como ejemplo, una página con un elemento h2 seguido de varias subsecciones marcadas con etiquetas h4 confundirá a un lector de pantalla. Con seis opciones, es tentador usar una etiqueta porque se ve mejor en un navegador, pero puede usar CSS para editar el tamaño relativo.

Un punto final, cada página siempre debe tener un (y solo un) elemento h1, que es el tema principal de tu contenido. Este y los otros títulos son utilizados en parte por los motores de búsqueda para comprender el tema de la página.

- ejercicio del Freecode-->Camper Cat quiere una página en su sitio dedicada a convertirse en un ninja. Ayúdelo a arreglar los títulos para que su lenguaje de marcado de un significado semántico al contenido y muestre las relaciones padre-hijo adecuadas de sus secciones. Cambia todas las etiquetas h5 al nivel de título adecuado para indicar que son subsecciones de las h2. Utiliza etiquetas h3 para este propósito.

{Lección 4} Salta directamente al contenido usando el elemento principal (main):

HTML5 introdujo varios elementos nuevos que dan a los desarrolladores más opciones y al mismo tiempo incorporan características de accesibilidad. Estas etiquetas incluyen main, header, footer, nav, article, y section, entre otros.

De forma predeterminada, un navegador muestra estos elementos de forma similar al humilde div. Sin embargo, usarlos cuando sea apropiado le da un significado adicional a tu lenguaje de marcado. El nombre de la etiqueta solo puede indicar el tipo de información que contiene, lo que agrega significado semántico a ese contenido. Las tecnologías de asistencia pueden acceder a esta información para proporcionar mejores opciones de resumen de páginas o de navegación a sus usuarios.

El elemento main se usa para envolver (lo adivinaste) el contenido principal, y solo debe haber uno por página. Su propósito es rodear la información relacionada con el tema central de tu página. No está destinado a incluir elementos que se repiten en todas las páginas, como enlaces de navegación o banners.

La etiqueta main también tiene una característica de referencia incrustada que la tecnología de asistencia puede utilizar para navegar al contenido principal rápidamente. Si alguna vez has visto un enlace de "Saltar al contenido principal" en la parte superior de la página, el uso de la etiqueta main proporciona automáticamente esa funcionalidad a los dispositivos de asistencia.

- ejercicio del freecode-->Camper Cat tiene algunas grandes ideas para su página de armas ninja. Ayúdelo a configurar su marcado agregado etiquetas de apertura y cierre main entre el header y el footer (cubierto en otros desafíos). Mantenga las etiquetas main vacías por ahora.

{Lección 5} Envolver el contenido en el elemento article:

article es otro de los nuevos elementos de HTML5 que añaden significado semántico a su marcado. article es un elemento seccionador y se utiliza para envolver contenido independiente y autónomo. La etiqueta funciona bien con entradas de blog, publicaciones en el foro o artículos de noticias.

Determinar si el contenido puede ser independiente suele ser una cuestión de criterio, pero puedes hacer un par de pruebas simples. Pregúntate, si elimino todo el contexto circundante, ¿ese contenido aún tendría sentido? Del mismo modo, para el texto, ¿se mantendría el contenido si estuviera en una fuente RSS?

Recuerda que las personas que usan tecnologías de asistencia dependen de un lenguaje de marcado organizado y semánticamente significativo para comprender mejor su trabajo.

Nota: El elemento *section* también es nuevo HTML5, y tiene significado semántico ligeramente diferente al de *article*. Un article es para contenido independiente, y una section es para agrupar contenido relacionado temáticamente. Se pueden usar uno dentro del otro, según sea necesario. Por ejemplo, si un libro es el *article*, entonces cada capítulo es una *section*. Cuando no haya relación entre grupos de contenido, usa un div.

* - grupos de contenido <section> - grupos de contenido relacionado <article> - grupos independientes, contenido autónomo*

- ejercicio del freecode-->Camper Cat usó etiquetas article para envolver las publicaciones en la página de su blog, pero olvido usarlas en la parte superior. Cambia la etiqueta div para usar una etiqueta article.

{Lección 6} Haz que la navegación del lector de pantalla sea más fácil con el encabezado (Header) Landmark:

El siguiente elemento HTML5 que agrega significado semántico y mejora la accesibilidad es la etiqueta header. Se usa para envolver información introductoria o enlaces de navegación para su etiqueta principal y funciona bien con el contenido que se repite en la parte superior en varias páginas.

header comparte la función landmark integrada que viste con main, lo que permite a las tecnologías de asistencia navegar rápidamente a ese contenido.

Nota: El header está diseñado para usarse dentro de la etiqueta body de tu documento HTML. Es diferente al elemento head, que contiene el título de la página, la meta información, etc.

- ejercicio del freecode --> Camper Cat está escribiendo algunos grandes artículos sobre el entrenamiento ninja, y quiere añadir una página para ellos a su sitio. Cambia la parte superior div que actualmente contiene el h1 a una etiqueta header.

{Lección 7} Haz que la navegación del lector de pantalla sea más fácil con el nav Landmark:

El elemento nav es otro elemento de HTML5 con la función de punto de referencia insertado para facilitar la navegación del lector de pantalla. Esta etiqueta esta destinada a envolver los principales enlaces de navegación en tu página.

Si hay enlaces internos repetidos en la parte de la página, no es necesario usar el lenguaje de marcado con aquellos con una etiqueta nav también. Usando una etiqueta footer (cubierto en el próximo desafío) es suficiente.

- ejercicio del freecode-->Camper Cat incluyó enlaces de navegación en la parte superior de su página de entrenamiento, pero los envolvió en un div. Cambia la etiqueta div a una etiqueta nav para mejorar la accesibilidad en su página.

{Lección 8} Haz que la navegación del lector de pantalla sea más fácil con el footer Landmark:

Similar a header y nav, el elemento footer tiene una característica de referencia incorporada que permite a los dispositivos de asistencia navegar rápidamente hacia él. Se utiliza principalmente para contener información sobre derechos de autor o enlaces a documentos relacionados que normalmente se encuentran en la parte inferior de una página.

- ejercicio del freecode-->La página de entrenamiento del Camper Cat está progresando bien. Cambia el div que utilizó para envolver su información de copyright en la parte inferior de la página a un elemento footer.

{Lección 9} Mejorar la accesibilidad del contenido de audio con el elemento de audio:

El elemento audio de HTLM le da un significado semántico cuando contiene sonido o contenido de flujo de audio en el código. El contenido del audio también necesita un texto alternativo para que las personas sordas o con dificultad para escuchar puedan acceder al mismo. Para esto debe tener un texto cercano en la página o un enlace a una transcripción.

La etiqueta de audio suporta los atributos controls. Esto muestra los controles por defecto de reproducir, pausar, entre otros controles, y soporta la funcionalidad del teclado. Este es un atributo booleano por lo que no necesita un valor, su presencia en la etiqueta activa la configuración.

Acá tenemos un ejemplo

```
<audio id="meowClip" controls>
  <source src="audio/meow.mp3" type="audio/mpeg">
  <source src="audio/meow.ogg" type="audio/ogg">
</audio>
```

Note: El contenido multimedia generalmente tiene componentes visuales y de audio. Se necesita sincronizar los subtítulos y una transcripción para que los usuarios con dificultades de vista o con problemas para escuchar puedan tener acceso a las mismas. Por lo general, un desarrollador de web no se responsabiliza de la creación de subtítulos o transcripciones pero necesita saber para incluirlos.

- ejercicio del freecode-->Es hora de salir del Camper Cat y conocer a nuestro compañero Zersiax (@zersiax), un campeón en accesibilidad y un usuario de lector de pantalla. Para escuchar un clip de su lector de pantalla en acción, añade un elemento audio después de p. Incluye el atributo controls. A continuación, coloca una etiqueta source dentro de las etiquetas audio con el atributo src establecido en https://s3.amazonaws.com/freecodecamp/screen-reader.mp3 y el atributo type establecido en "audio/mpeg".
  Note: El clip del audio puede sonar rápido y quizás sea difícil de entender pero esta velocidad es normal para los lectores de pantalla.

{Lección 10} Mejorar la accesibilidad de gráficos con el elemento figure:

HTML5 introdujo el elemento figure y el figcaption relacionado. Cuando se usan juntos, estos elementos envuelven una representación visual (como puede ser una imagen, diagrama o gráfico) junto con su leyenda. Envolviendo estos elementos da un impulso de accesibilidad doble al agrupar semánticamente el contenido relacionado y proporciona una alternativa de texto explicando la figura figure.

En visualizaciones de datos como gráficos, la leyenda o "figcaption" puede ser utilizada para resumir en formato de texto las tendencias o conclusiones, para beneficio de usuarios con discapacidades visuales. Otro de los desafíos se ocupa cómo mover fuera de la pantalla una versión en formato de tabla con los datos del gráfico (usando CSS) para ayudar a usuarios de lectores de pantalla.

Aquí hay un ejemplo: Ten en cuenta que el elemento figcaption va dentro de las etiquetas figure y se puede combinar con otros elementos

<figure>

<img src="roundhouseDestruction.jpeg" alt="Photo of Camper Cat executing a roundhouse kick">

*
*

<figcaption>

Master Camper Cat demonstrates proper form of a roundhouse kick.

</figcaption>

</figure>

- ejercicio del freecode-->Camper Cat está trabajando para crear un gráfico de barras apiladas que muestre la cantidad de tiempo semanal a dedicar en entrenamiento en sigilo, combate y armas. Ayúdalo a estructurar mejor su página cambiando la etiqueta div que usó por una etiqueta figure, y cambiando la etiqueta p que rodea la leyenda por una etiqueta figcaption.

{Lección 11} Mejorar la accesibilidad del campo de formulario con el elemento label (etiqueta):

La mejora de la accesibilidad con el marcado semántico HTML se aplica al uso de nombres de etiquetas y atributos apropiados. Los próximos desafíos cubren algunos escenarios importantes utilizando atributos en formularios.

La etiqueta label contiene el texto para un elemento específico de control de formulario, por lo general el nombre o etiqueta para una elección. Esto vincula el significado al elemento y hace que el formulario se lea mejor. El atributo for en una etiqueta label asocia de manera explícita dicho label con el control de formulario utilizado por los lectores de pantalla.

Ya aprendiste sobre botones de radio y sus etiquetas en una lección de la sección de HTML básico. En esa lección, colocamos el elemento de entrada del botón de radio dentro de la etiqueta label junto con la etiqueta del texto para hacer que el texto se pueda cliquear. Otra forma de lograr esto es usando el atributo for, como se explica en esta lección.

El valor del atributo for debe ser igual al valor del atributo id del formulario de control. Por ejemplo

```
<form>
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
</form>
```

- ejercicio del freecode-->Camper Cat espera que haya mucho interés en sus publicaciones bien armadas en el blog y quiere incluir un formulario de registro por correo electrónico. Añade un atributo for en el correo electrónico label que coincida con el id en su campo input.

{Lección 12} Envolver los botones de radio en un elemento fieldset para una mejor accesibilidad:

El siguiente tema sobre formularios cubre la accesibilidad de los botones tipo radio. A cada opción se le da una label (etiqueta) con un atributo for vinculado al id del elemento correspondiente como se cubrió en el último desafío. Dado que los botones de radio a menudo vienen en un grupo donde el usuario debe elegir uno, hay una manera de mostrar semánticamente que las opciones son parte de un conjunto.

La etiqueta fieldset rodea toda la agrupación de botones de radio para lograr esto. A menudo utiliza una etiqueta de legend para proporcionar una descripción para la agrupación, que lectores de pantalla leen por cada opción en el elemento fieldset.

El contenedor fieldset y la etiqueta legend no son necesarias cuando las opciones se explican por sí mismas, como selección de género. Usar label con el atributo for para cada botón de radio es suficiente.

Aquí hay un ejemplo

```
<form>
 <fieldset>
  <legend>Choose one of these three items:</legend>
  <input id="one" type="radio" name="items" value="one">
  <label for="one">Choice One</label><br>
  <input id="two" type="radio" name="items" value="two">
  <label for="two">Choice Two</label><br>
  <input id="three" type="radio" name="items" value="three">
  <label for="three">Choice Three</label>
 </fieldset>
</form>
```

- ejercicio del freecode-->Camper Cat quiere información sobre el nivel ninja de sus usuarios cuando se registran en su lista de correo electrónico. Agregó un conjunto de botones de radio y aprendió de nuestra última lección a usar etiquetas label con atributos for para cada opción. ¡Vamos Camper Cat! Sin embargo, su código todavía necesita ayuda. Cambia la etiqueta div que rodea los botones de radio a una etiqueta fieldset y cambia la etiqueta p dentro de ella a una legend.

{Lección 13} Agrega un selector de fechas (date) accesible:

Los formularios suelen incluir el campo input, que puede usarse para crear diferentes tipos de controles en los formularios. El atributo type en este elemento indica el tipo de elemento input a crear.

Puede que hayas visto los tipos de campo text y submit en desafíos anteriores. HTML5 además introdujo una opción para especificar un campo date para fechas. Dependiendo del soporte de los navegadores, un selector de fechas debería aparecer cuando el campo input esté en foco, y esto hace mucho más sencillo para los usuarios cargar información en el formulario.

Para los navegadores más antiguos, el tipo será por defecto text, por lo que ayuda a mostrar a los usuarios el formato de fecha(date) esperado en el texto label o placeholder por si acaso.

Aquí hay un ejemplo

<label for="input1">Enter a date:</label>

<input type="date" id="input1" name="input1">

- ejercicio del freecode-->Camper Cat está organizando un torneo de Mortal Kombat y quiere pedir a los participantes que consideren cuál fecha les resultaría mejor. Agrega una etiqueta input con un atributo type de date, un atributo id de pickdate y un atributo name con valor date.

{Lección 14} Estandarizar horas con el atributo HTML5 datetime:

Continuando con el tema de fechas, HTML5 también introdujo el elemento time junto con un atributo datetime para estandarizar las horas. El elemento time es un elemento en línea que puede ajustar una fecha u hora en una página. Un atributo datetime contiene un formato válido para esa fecha. Este es el valor al que acceden los dispositivos de asistencia. Ayuda a evitar la confusión al declarar una versión estandarizada de un tiempo, incluso si está escrita informal o coloquialmente en el texto.

Aquí hay un ejemplo

* Master Camper Cat officiated the cage match between Goro and Scorpion <time datetime="2013-02-13">last Wednesday</time>, which ended in a draw. *

- ejercicio del freecode-->¡Ya tenemos los resultados de la encuesta de Mortal Kombat de Camper Cat! Envuelve una etiqueta time alrededor del texto Thursday, September 15<sup>th<sup> y agrega un atributo datetime establecido en 2016-09-15.

{Lección 15} Hacer que los elementos solo sean visibles para un lector de pantalla mediante CSS personalizado:

¿Has notado que todos los desafíos de accesibilidad aplicados hasta ahora no han usado ningún CSS? Esto muestra la importancia de utilizar un esquema de documento lógico y etiquetas semánticamente significativas alrededor de su contenido antes de introducir el aspecto de diseño visual.

Sin embargo, la magia de CSS también puede mejorar la accesibilidad en tu página cuando deseas ocultar visualmente contenido destinado solo para lectores de pantalla. Esto sucede cuando la información está en un formato visual (como un gráfico), pero los usuarios del lector de pantalla necesitan una presentación alternativa (como una tabla) para acceder a los datos. CSS se utiliza para colocar los elementos exclusivos de lector de pantalla fuera del área visual de la ventana del navegador.

Aquí hay un ejemplo de las reglas de CSS que logran esto

```
.sr-only {
  position: absolute;
  left: -10000px;
  width: 1px;
  height: 1px;
  top: auto;
  overflow: hidden;
}
```

Nota: Los siguientes enfoques CSS No harán lo mismo

display: none; o *visibility: hidden;* oculta el contenido para todos, incluidos los usuarios del lector de pantalla

Los valores cero para los tamaños del píxel, como width: 0px; height: 0px; eliminan ese elemento del flujo de tu documento, lo que significa que los lectores de pantalla lo ignorarán

- ejercicio del freecode-->Camper Cat creó un gráfico de barras apiladas realmente genial para su página de entrenamiento y coloco los datos en una tabla para sus usuarios con dificultad visual. La tabla ya tiene una clase sr-only, pero las reglas de CSS aún no se han completado. Asigna a position un valor absolute, a left un valor de -10000px, y tanto a width como a height un valor de 1px.

{Lección 16} Mejorar la legibilidad con texto de alto contraste:

El bajo contraste entre los colores de primer plano y de fondo puede dificultar la lectura del texto. Suficiente contraste mejora la legibilidad de tu contenido, pero ¿qué significa exactamente "suficiente"?

Las Directrices de Accesibilidad al Contenido Web (WCAG) recomiendan al menos una relación de contraste de 4.5 a 1 para el texto normal. La relación se calcula comparando los valores de luminancia relativa de dos colores. Esto varía de 1:1 para el mismo color, o ningún contraste, a 21:1 para blanco contra negro, el contraste más sustancial. Hay muchas herramientas de comprobación de contraste disponibles en línea que calculan esta relación por ti.

- ejercicio del freecode-->La elección del Gato Campero de texto gris claro sobre un fondo blanco para su reciente publicación de blog tiene una relación de contraste de 1.5:1, por lo que es difícil de leer. Cambie el color del texto del gris, actual (#D3D3D3) a un gris más oscuro (#636363) para mejorar la relación de contraste a 6:1.

{Lección 17} Evitar problemas de percepción del color usando el suficiente contraste:

El color es una parte importante del diseño visual, pero su aplicación presenta dos problemas de accesibilidad. El primer problema es que no se debe utilizar el color como la única forma de transmitir información importante, porque los lectores de pantalla no lo distinguen. En segundo lugar, los colores de primer plano y de fondo necesitan tener suficiente contraste para que los usuarios sean capaces de distinguirlos.

Los desafíos anteriores explicaron la existencia de texto alternativo para remediar el primer problema. El último desafío introdujo herramientas de comprobación de contraste para ayudar con el segundo problema. El índice de contraste de 4.5:1 recomendado por WCAG se aplica tanto para los colores como para combinaciones de escalas de grises.

Los usuarios daltónicos tienen problemas para distinguir algunos colores de otros. Esto generalmente depende del tono pero también depende a veces de su luminosidad. Posiblemente recuerdes que el índice de contraste se calcula utilizando los valores de luminancia (o luminosidad) relativa de los colores de primer plano y del fondo.

En la práctica, la relación de contraste de 4.5:1 puede alcanzarse oscureciendo el color más oscuro (o sea, añadiendo negro) y aclarando el color más claro (añadiéndole blanco). Se considera que los tonos más oscuros en la rueda de color son los azules, violetas, magentas y los rojos, mientras que los colores más claros son los naranjas, amarillos, verdes y azul-verdosos.

- ejercicio del freecode-->Camper Cat está experimentando con usar color en el texto y fondo de su blog, pero su combinación actual de background-color verdoso con color de texto granate tiene un índice de contraste de 2.5:1. Puedes ajustar fácilmente la luminosidad de los colores, ya que los declaró usando la propiedad CSS hsl() (que significa tono, saturación, luminosidad o "hue, saturation, lightness") cambiando solo el tercer argumento. Aumenta el valor de luminosidad del background-color de 35% a 55%, y disminuye el valor de luminosidad del color del texto del 20% al 15%. Esto mejora el índice de contraste llevándolo a 5.9:1.

{Lección 18} Evitar problemas de color para usuarios daltónicos eligiendo cuidadosamente los colores que transmiten información:

Existen varias formas de daltonismo o ceguera al color. Estos pueden ir desde tener una sensibilidad reducida a una longitud de onda de luz específica, hasta la incapacidad total de ver algún color. La forma más común en que se presenta es una sensibilidad reducida para detectar tonos de verde.

Por ejemplo, si dos colores verdes similares son el color usado en el primer plano y el fondo de tu contenido, un usuario daltónico podría no ser capaz de distinguirlos entre sí. Podemos pensar en los colores cercanos como aquellos colores que son vecinos o adyacentes en la rueda de color. Por este motivo, las combinaciones de estos colores deben evitarse cuando se usan para transmitir información importante.

Nota: Algunas herramientas de selección de colores disponibles en Internet incluyen simulaciones visuales de cómo serían vistos por usuarios con diversos tipos de daltonismo. Estos son excelentes recursos que puedes aprovechar, sumados a las calculadoras para verificar el contraste de color que puedes encontrar en Internet.

ejercicio del freecode-->Camper Cat está probando diferentes estilos para un botón importante, pero el background-color amarillo (#FFF33) y el color de texto verde (#33FF33) son

tonos vecinos en la rueda de color, por lo que resultan prácticamente indistinguibles para algunos usuarios daltónicos. (Además, como tienen un nivel de luminosidad similar, no pasan la verificación de índice de contraste o contrast ratio). Cambia el color del texto a azul oscuro (#003366) para resolver ambos problemas.

{Lección 19} Dar significado a los enlaces agregando un texto descriptivo:

Los lectores de pantalla tienen varias opciones para qué tipo de contenido lee su dispositivo. Estas opciones incluyen saltar a elementos de referencia (o sobre) o saltar al contenido principal, u obtener un resumen de la página en los títulos. Otra opción es escuchar la narración de los enlaces disponibles en una página.

Los lectores de pantalla hacen esto leyendo el texto del enlace, o lo que haya entre las etiquetas anchor (a). Tener una lista de enlaces que solo digan "clic aquí" o "leer más" no ayuda. En su lugar, utilice texto breve pero descriptivo dentro de las etiquetas a para proporcionar más significado a estos usuarios.

- ejercicio del freecode-->El texto de los enlaces que utiliza el Camper Cat no es muy descriptivo si se lo toma fuera de su contexto. Mueve las etiquetas anchor (a) para que envuelvan el texto information about batteries en lugar de Click here.

{Lección 20} Hacer que los enlaces sean navegables con claves de acceso HTML:

HTML ofrece el atributo accesskey para especificar una tecla de acceso directo para activar o enfocar un elemento. Añadiendo un atributo accesskey puede hacer que la navegación sea más eficiente para los usuarios que solo utilizan teclado.

HTML5 permite que este atributo se use en cualquier elemento, pero es particularmente útil cuando se usa con elementos interactivos. Esto incluye enlaces, botones y controles de formulario.

Aquí hay un ejemplo

<button accesskey="b">Important Button</button>

- ejercicio del freecode-->Camper Cat quiere que los enlaces alrededor de los dos títulos de artículos de blog tengan atajos de teclado para que los usuarios de su sitio puedan navegar rápidamente a la historia completa. Agrega un atributo accesskey a ambos enlaces y establece el primero en g (para Garfield) y el segundo en c (para Chuck Norris).

{Lección 21} Usa tabindex para agregar enfoque de teclado a un elemento:

El atributo HTML tabindex tiene tres funciones distintas relacionadas con el foco del teclado de un elemento. Cuando está en una etiqueta, indica que se puede hacer foco en el elemento. El valor (un número entero que es positivo, negativo o cero) determina el comportamiento.

Ciertos elementos, como los vínculos y los controles de formulario, reciben automáticamente el foco del teclado cuando un usuario pestañas a través de una página. Está en el mismo orden en que los elementos vienen en la fuente del lenguaje de marcado de HTML. Esta misma funcionalidad se puede dar a otros elementos, como div, span y p, colocando un atributo tabindex="0". Aquí hay un ejemplo

* I need keyboard focus! *

nota: Un valor negativo de tabindex (normalmente -1) indica que un elemento es enfocable, pero no es accesible por el teclado. Este método generalmente se usa para enfocar el contenido mediante programación (como cuando se activa un div utilizando para una ventana emergente), y esta más allá del alcance de estos desafíos.

- ejercicio del freecode-->Camper Cat creó una nueva encuesta para recopilar información sobre sus usuarios. Él sabe que los campos de entrada obtienen automáticamente el enfoque del teclado, pero quiere asegurarse de que los usuarios de su teclado hagan una pausa en las instrucciones mientras tabulan los elementos. Agrega un atributo tabindex a la etiqueta p y establece su valor en 0. Extra: el uso de tabindex también permite que la pseudo-clase CSS :focus funcione en la etiqueta p.

{Lección 22} Utilizar tabindex para especificar el orden de enfoque del teclado para múltiples elementos:

El atributo tabindex también especifica el orden de tabulación exacto de los elementos. Esto se logra cuando el valor del atributo se establece en un número positivo de 1 o superior.

Establecer un tabindex="1" hará que el teclado se enfoque primero en ese elemento. Luego, recorre la secuencia de valores tabindex especificados (2, 3, etc.), antes de pasar a los elementos predeterminados y tabindex="0".

Es importante tener en cuenta que cuando el orden de tabulación se establece de esta manera, anula el orden predeterminado (que usa el código fuente HTML). Esto puede confundir a los usuarios que esperan comenzar la navegación desde la parte superior de la página. Esta técnica puede ser necesaria en algunas circunstancias pero en términos de accesibilidad, ten cuidado antes de aplicarla.

Aquí hay un ejemplo

* I get keyboard focus, and I get it first! *

* I get keyboard focus, and I get it second! *

ejercicio del freecode-->Camper Cat tiene un campo de búsqueda en su página de Citas Inspiradoras que planea colocar en la esquina superior derecha con CSS. Él quiere que los controles de formulario de búsqueda input y envío input sean los dos primeros elementos en el orden de tabulación. Agrega un atributo tabindex establecido en 1 al search input, y un atributo tabindex establecido en 2 al submit input.

[Principios de diseño web responsivo]

{Lección 1} Crear un media query:

Las consultas de medios (Media Queries) son una nueva técnica introducida en CSS3 que cambia la presentación de contenido basado en diferentes tamaños de viewport. El viewport es el área visible del usuario de una página web, y es diferente dependiendo del dispositivo utilizado para acceder al sitio.

Las consultas de medios se basan en un tipo de medio, y si ese tipo de medio coincide con el tipo de dispositivo en el que se muestra el documento, los estilos se aplican. Puedes tener tantos selectores y estilos dentro de tu consultas de medios como desees.

Este es un ejemplo de una consultas de medios que devuelve el contenido cuando el ancho del dispositivo es menor o igual a 100px

@media (max-width: 100px) { / CSS Rules */ }*

y la siguiente consultas de medios devuelve el contenido cuando la altura del dispositivo es mayor o igual a 350px

@media (min-height: 350px) { / CSS Rules */ }*

Recuerda, el CSS dentro de las consultas de medios se aplica sólo si el tipo de medio coincide con el del dispositivo que se está usando.

- ejercicio del freecode-->Agrega una consultas de medios, de forma que la etiqueta p tenga un font-size de 10px cuando la altura del dispositivo sea menor o igual a 800px.

{Lección 2} Hacer una imagen responsiva:

Hacer imágenes responsivas con CSS es realmente simple. Sólo tienes que agregar estas propiedades a una imagen

```
img {
  max-width: 100%;
  height: auto;
}
```

El ancho máximo max-width de 100% se asegurará de que la imagen nunca es más ancha que el contenedor en el que se encuentra. y la altura height de auto hará que la imagen mantenga su relación de aspecto original.

- ejercicio del freecode-->Agrega las reglas de estilo a la clase responsive-img para hacerla responsiva. Nunca debería ser más ancha que su contenedor (en este caso, es la ventana de vista previa) y debería mantener su relación de aspecto original. Después de haber agregado tu código, redimensiona la vista previa para ver cómo se comportan tus imágenes.

{Lección 3} Usar una imagen retina para pantallas de alta resolución:

Con el aumento de los dispositivos conectados a Internet, sus tamaños y especificaciones varían, y las pantallas que utilizan podrían ser diferentes tanto externa como internamente. La densidad de píxeles es un aspecto que puede ser diferente de un dispositivo a otro, esta densidad se conoce como Pixel Por Pulgada, en inglés "pixels per inch" (PPI) o Puntos por Pulgada, en inglés "dots per inch" (DPI). La pantalla más famosa que aprovecha esto es la conocida como "Pantalla Retina" en los últimos portátiles Apple MacBook Pro, y recientemente en los ordenadores iMac. Debido a la diferencia en la densidad de píxeles entre las pantallas de "Retina" y "No Retina", algunas imágenes que no han sido hechas con una pantalla de alta resolución en mente podrían verse "pixeladas" cuando se muestran en una pantalla de alta resolución.

La forma más sencilla de hacer que tus imágenes aparezcan correctamente en pantallas de alta resolución, tales como la "pantalla retina" de las MacBook Pros es definir sus valores de ancho width y de altura height como sólo la mitad de lo que es el archivo original. Aquí hay un ejemplo de una imagen que solo utiliza la mitad de la altura y ancho originales

<style>

img { height: 250px; width: 250px; }

</style>

<img src="coolPic500x500" alt="A most excellent picture">

- ejercicio del freecode-->Establece el width y height de la etiqueta img a la mitad de sus valores originales. En este caso, tanto el height original como el width original son de 200px.

{Lección 4} Hacer tipografía responsiva:

En lugar de usar em o px para dimensionar texto, puedes usar unidades de viewport para obtener una tipografía responsiva. Las unidades de viewport, como los porcentajes, son unidades relativas, pero se basan en objetos diferentes. Las unidades de viewport son relativas a las dimensiones del viewport (ancho o alto) de un dispositivo, y los porcentajes son relativos al tamaño del elemento contenedor padre.

Las cuatro diferentes unidades de viewport son:

vw (viewport width): 10vw sería el 10% del ancho del viewport.

vh (viewport height): 3vh sería el 3% del alto del viewport.

vmin (viewport mínimo): 70vmin sería el 70% de la dimensión más pequeña del viewport (altura o ancho).

vmax (viewport máximo): 100vmax sería el 100% de la dimensión más grande del viewport (altura o ancho).

Aquí hay un ejemplo que establece una etiqueta body al 30% del ancho del viewport.

body { width: 30vw; }

ejercicio del freecode-->Establece el ancho width de la etiqueta h2 al 80% del ancho del viewport y el ancho width del párrafo como el 75% de la dimensión más pequeña del viewport.

[CSS Flexbox]

{Lección 1} Utilizar display: flex para posicionar dos cajas:

Esta sección utiliza estilos de desafío alternos para mostrar cómo usar CSS para posicionar elementos de una manera flexible. En primer lugar, un desafío explicará la teoría, luego un desafío práctico utilizando un simple componente de tweet aplicará el concepto de flexbox.

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

- ejercicio del freecode-->Agrega la propiedad CSS display a #box-container y establece su valor como flex.

{Lección 2} Agregar superpoderes flex al tweet incrustado:

A la derecha está el tweet incrustado que se utilizará como ejemplo práctico. Algunos de los elementos lucirían mejor con una disposición diferente. El último desafío demostró display: flex. Aquí la agregaras a varios componentes en el tweet incrustado para empezar a ajustar su posición.

- ejercicio del freecode-->Agrega la propiedad CSS display: flex a todos los siguientes elementos (ten en cuenta que los selectores ya están configurados en el CSS):

El encabezado header, el .profile-name del encabezado, el .follow-btn del encabezado, el h3 y h4 del encabezado, el footer, y el .stats del pie de página (footer).

{Lección 3} Utilizar 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.

- ejercicio del freecode-->Agrega la propiedad CSS flex-direction al elemento #box-container y asígnale un valor de row-reverse.

{Lección 4} Aplicar la propiedad flex-direction para crear filas en el tweet Insertado:

El header y footer en el ejemplo de tweet insertado tienen elementos secundarios que podrían organizarse como filas usando la propiedad flex-direction. Esto le dice a CSS que alinee los hijos horizontalmente.

- ejercicio del freecode-->Agrega la propiedad CSS flex-direction tanto al header como al footer y establece el valor en row.

{Lección 5} Utilizar la propiedad flex-direction para hacer una columna:

Los dos últimos desafíos usaron la propiedad flex-direction establecida en row. Esta propiedad también puede crear una columna apilando verticalmente los hijos de un contenedor flex.

- ejercicio del freecode-->Agrega la propiedad CSS flex-direction al elemento #box-container y asígnale un valor de column.

{Lección 6} Aplicar la propiedad flex-direction para crear columnas en el tweet insertado:

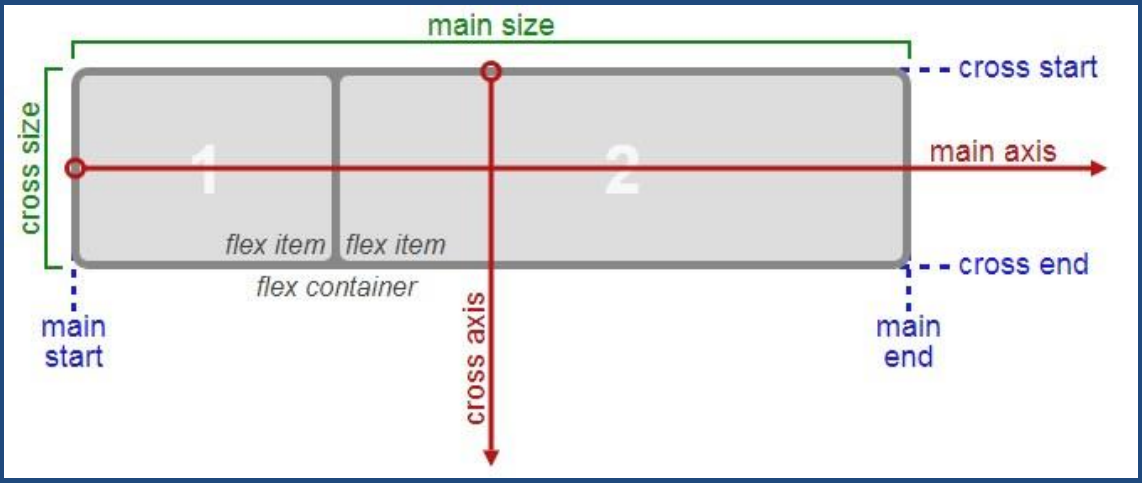
El header y footer del tweet insertado usaron previamente la propiedad flex-direction con un valor de fila. De manera similar, los elementos dentro del elemento .profile-name funcionarían bien apilados como una columna.

- ejercicio del freecode-->Agrega la propiedad CSS flex-direction al elemento .profile-name del título y establezca el valor en column.

{Lección 7} Alinear 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.

Aquí hay una imagen útil que muestra una fila para ilustrar los conceptos siguientes.



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 elementos flexibles con un espacio completo en ambos extremos del contenedor flexible

- ejercicio del freecode-->Un ejemplo ayuda a mostrar esta propiedad en acción. Agrega la propiedad CSS justify-content al elemento #box-container y asígnale un valor de center.
  Extra: Prueba las otras opciones de la propiedad justify-content en el editor de código para ver sus diferencias. Pero ten en cuenta que un valor de center es el único que superará este desafío.

{Lección 8} Utilizar la propiedad *justify-content* en el tweet Insertado:

El último desafío mostró un ejemplo de la propiedad justify-content. Para el tweet insertado, esta propiedad se puede aplicar para alinear los elementos en el .profile-name.

- ejercicio del freecode-->Agrega la propiedad CSS justify-content al .profile-name y establece el valor en cualquiera de las opciones del último desafío.

{Lección 9} Alinear 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.

- ejercicio del freecode-->Un ejemplo ayuda a mostrar esta propiedad en acción. Agrega la propiedad CSS align-items al elemento #box-container y asígnale un valor de center. Extra: Prueba las otras opciones de la propiedad align-items en el editor de código para ver sus diferencias. Pero ten en cuenta que un valor de center es el único que superará este desafío.

{Lección 10} Utilizar la propiedad align-items en el tweet insertado:

El último desafío introdujo la propiedad align-items y dio un ejemplo. Esta propiedad se puede aplicar a unos cuantos elementos del tweet insertado para alinear los elementos flexibles dentro de ellos.

- ejercicio del freecode-->Agrega la propiedad CSS align-items al elemento .follow-btn del encabezado. Establece el valor a center.

{Lección 11} Usar la propiedad flex-wrap para envolver una fila o columna:

CSS flexbox tiene una característica para dividir un elemento flexible 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.

- ejercicio del freecode--> La disposición actual tiene demasiadas cajas para una sola fila. Agrega la propiedad CSS flex-wrap al elemento #box-container y asígnale un valor de wrap.

{Lección 12} 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.

- ejercicio del freecode-->Agrega la propiedad CSS flex-shrink tanto a #box-1 como a #box-2. Da a #box-1 un valor de 1 y a #box-2 un valor de 2.

{Lección 13} Usar 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.

ejercicio del freecode-->Agrega la propiedad CSS flex-grow tanto a #box-1 como a #box-2. Da a #box-1 un valor de 1 y a #box-2 un valor de 2.

{Lección 14} Usar la propiedad *flex-basis* para establecer el tamaño inicial de un elemento:

La propiedad flex-basis especifica el tamaño inicial del elemento antes de que CSS haga ajustes con flex-shrink o flex-grow.

Las unidades usadas por la propiedad flex-basis son las mismas que otras propiedades de tamaño (px, em, %, etc.). El valor auto dimensiona los elementos basándose en el contenido.

- ejercicio del freecode-->Establece el tamaño inicial de las cajas usando flex-basis. Agrega la propiedad CSS flex-basis tanto a #box-1 como a #box-2. Da a #box-1 un valor de 10em y a #box-2 un valor de 20em.

{Lección 15} Usar la propiedad abreviada *flex*:

Hay un atajo disponible para establecer varias propiedades flex a la vez. Las propiedades flex-grow, flex-shrink, y flex-basis pueden establecerse utilizando la propiedad flex.

Por ejemplo, *flex: 1 0 10px*; establecerá las propiedades del elemento en *flex-grow: 1;, flex-shrink: 0;, y flex-basis: 10px;.*

La configuración predeterminada de la propiedad es *flex: 0 1 auto;.*

- ejercicio del freecode-->Agrega la propiedad CSS flex tanto a #box-1 como a #box-2. Dale a #box-1 los valores para que su flex-grow sea 2, su flex-shrink sea 2, y su flex-basis sea 150px. Dale a #box-2 los valores para que su flex-grow sea 1, su flex-shrink sea 1, y su flex-basis sea 150px.
  Estos valores causarán que para llenar el espacio extra #box-1 crezca el doble de #box-2 cuando el contenedor sea mayor que 300px y se reduzca al doble de #box-2 cuando el contenedor sea menor de 300px. 300px es el tamaño combinado de los valores de flex-basis de las dos cajas.

{Lección 16} Usar la propiedad *order* para reorganizar los elementos:

La propiedad *order* se utiliza para indicarle a CSS el orden en que aparecen los elementos flexibles en el contenedor flex. Por defecto, los elementos aparecerán en el mismo orden que

vienen en el HTML de origen. La propiedad toma números como valores, y se pueden usar números negativos.

- ejercicio del freecode-->Agrega la propiedad CSS order tanto a #box-1 como a #box-2. Da a #box-1 un valor de 2 y a #box-2 un valor de 1.

{Lección 17} Usar la propiedad *align-self*:

La última propiedad para elementos flexibles es *align-self*. Esta propiedad te permite ajustar la alineación de cada elemento individualmente, en lugar de ajustarlos todos a la vez. Esto es útil ya que otras técnicas comunes de ajuste usan las propiedades CSS float, clear, y vertical-align, las cuales no funcionan en elementos flexibles.

align-self acepta los mismos valores que align-items y reemplazará cualquier valor establecido por la propiedad align-items.

- ejercicio del freecode-->Agrega la propiedad CSS align-self tanto a #box-1 como a #box-2. Da a #box-1 un valor de center y a #box-2 un valor de flex-end.

[CSS Grid]

{Lección 1} Crear tu primera CSS Grid:

Convierte cualquier elemento HTML en una grid al establecer la propiedad display a grid. Esto te da la habilidad de usar todas las demás propiedades asociadas con CSS Grid.

Nota: en CSS Grid, el elemento padre se refiere como el contenedor y sus hijos se llaman elementos.

- ejercicio del freecode-->Cambia la propiedad display del div que tiene la clase container a grid.

{Lección 2} Agrega columnas con grid-template-columns

Crear un simple elemento cuadrícula (grid) no te llevará muy lejos. Necesitas también definir su estructura. Para agregar columnas a la cuadrícula, usa la propiedad grid-template-columns en el contenedor de la cuadrícula como se demuestra a continuación

```
.container {
  display: grid;
  grid-template-columns: 50px 50px;
}
```

Esto le dará a tu cuadrícula dos columnas que tienen 50px de ancho cada una. El número de parámetros que se le da a la propiedad grid-template-columns indica el número de columnas en la cuadrícula y el valor de cada parámetro indica el ancho de cada columna.

- Ejercicio del freecode→Haz que el contenedor de la cuadrícula tenga tres columnas con un ancho de 100px cada una.

{Lección 3} Agregar filas con *grid-template-rows*:

La cuadrícula (grid) que creaste en el último desafío establecerá el número de filas automáticamente. Para ajustar las filas manualmente, usa la propiedad grid-template-rows de la misma manera en la que usaste grid-template-columns en el desafío anterior.

- ejercicio del freecode--> Agrega dos filas a la cuadrícula que tengan 50px de alto cada una.

{Lección 4} Usar unidades CSS Grid para cambiar el tamaño de las columnas y filas:

Puedes usar unidades absolutas y relativas como px y em en CSS Grid para definir el tamaño de filas y columnas. Puedes usar estas también:

fr: fija la columna o fila a una fracción del espacio disponible,

auto: fija la columna o fila al ancho o alto de su contenido automáticamente,

%: ajusta la columna o fila al porcentaje de ancho de su contenedor.

A continuación el código que genera el resultado de la vista previa

grid-template-columns: auto 50px 10% 2fr 1fr;

Esta línea de código genera cinco columnas. La primera columna es tan ancha como su contenido, la segunda tiene 50px de ancho, la tercera es el 10% de su contenedor y para las últimas dos columnas; el espacio restante es dividido en tres secciones: dos son asignadas a la cuarta columna y una a la quinta columna.

- ejercicio del freecode-->Haz una cuadrícula con tres columnas que tengan las siguientes dimensiones de ancho: 1fr, 100px y 2fr.

{Lección 5} Crear un espacio entre columnas usando *grid-column-gap*:

Hasta ahora en las grids que has creado, las columnas han estado todas juntas. Algunas veces querrás un espacio entre las columnas. Para agregar un espacio entre las columnas, usa la propiedadgrid-column-gap de la siguiente manera

grid-column-gap: 10px;

Esto crea espacios vacíos de 10px entre todas las columnas.

- ejercicio del freecode-->Haz que las columnas en la cuadrícula (grid) tengan 20px de espacio.

{Lección 6} Crear un espacio entre filas usando *grid-row-gap*:

Puedes agregar un espacio entre las filas usando grid-row-gap de la misma manera en la que agregaste un espacio entre las columnas en el desafío anterior.

- ejercicio del freecode-->Crea un espacio para las filas que tenga 5px de alto.

{Lección 7} Agregar espacios más rápido con *grid-gap*:

grid-gap es una propiedad abreviada para grid-row-gap y grid-column-gap de los dos desafíos anteriores que es más conveniente usar. Si grid-gap tiene un valor, creará un espacio entre todas las filas y columnas. Sin embargo, si hay dos valores, usará el primero de estos para poner los espacios entre las filas y el segundo para los espacios entre las columnas.

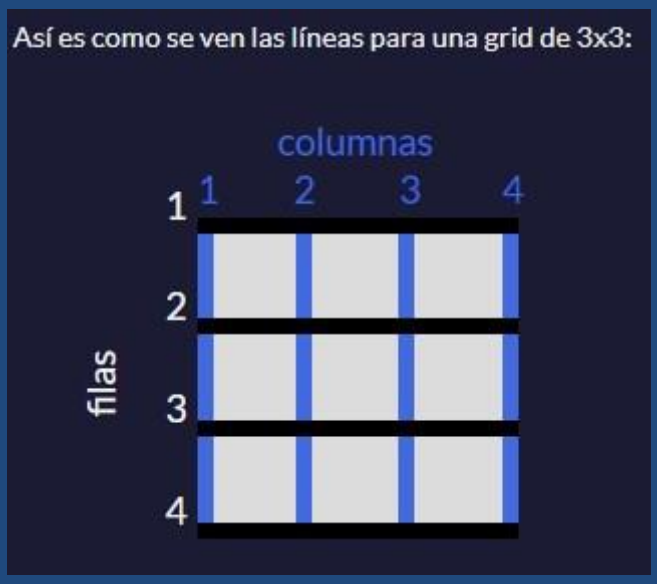
- ejercicio del freecode-->Usa grid-gap para introducir un espacio de 10px entre las filas y un espacio de 20px entre las columnas.

{Lección 8} Usar *grid-column* para controlar espaciado:

Hasta este punto, todas las propiedades que hemos discutido son para los contenedores de cuadrícula (grid). La propiedad grid-column es la primera que usaremos para los propios elementos de la cuadrícula.

Las líneas horizontales y verticales hipotéticas que crean la cuadrícula son referidas como líneas. Estas líneas son enumeradas empezando con el 1 desde la esquina superior izquierda de la cuadrícula y se desplaza hacia la derecha para las columnas y hacia abajo para las filas, contando hacia arriba.

Así es como se ven las líneas para una grid de 3x3:



Para controlar la cantidad de columnas que un elemento tendrá, puedes usar la propiedad grid-column en conjunto con el número de las líneas en las que quieres que empiece y termine.

Aquí un ejemplo

grid-column: 1 / 3;

Esto hará que el elemento empiece en la primera línea vertical de la grid a la izquierda y se extienda a la 3ra línea de la grid, ocupando dos columnas.

- ejercicio del freecode-->Haz que el elemento con clase item5 ocupe las dos últimas columnas de la cuadrícula.

{Lección 9} Usar *grid-row* para controlar espaciado:

Por supuesto, puedes hacer que los elementos ocupen múltiples filas así como se puede hacer con las columnas. Puedes definir las líneas horizontales donde quieres que un elemento empiece y termine usando la propiedad grid-row sobre un elemento de cuadrícula (grid).

- ejercicio del freecode-->Haz que el elemento con clase item5 ocupe las últimas dos filas.

{Lección 10} Alinear un elemento horizontalmente usando *justify-self*:

En CSS Grid, el contenido de cada elemento está ubicado en una caja, la cual se refiere como celda. Puedes alinear la posición del contenido dentro de la celda horizontalmente usando la propiedad justify-self en un elemento grid. Por defecto, esta propiedad tiene valor de stretch, lo que hace que el contenido cubra todo el ancho de la celda. Esta propiedad de CSS Grid acepta otros valores también

start: alinea el contenido hacia la izquierda de la celda,

center: alinea el contenido en el centro de la celda,

end: alinea el contenido hacia la derecha de la celda.

- ejercicio del freecode-->Usa la propiedad justify-self para centrar el elemento con la clase item2.

{Lección 11} Alinear un elemento verticalmente usando *align-self*:

Así como puedes alinear un elemento horizontalmente, hay una manera de alinearlo verticalmente también. Para hacer esto, usa la propiedad align-self sobre un elemento. Esta propiedad acepta los mismos valores que justify-self del desafío anterior.

- ejercicio del freecode-->Alinea verticalmente el elemento con clase item3 hacia el final end.

{Lección 12} Alinear todos los elementos horizontalmente usando *justify-items*:

Algunas veces querrás que todos los elementos en tu CSS Grid compartan el mismo alineamiento. Puedes usar las propiedades aprendidas anteriormente y alinearlos individualmente, o puedes alinear todos a la vez horizontalmente usando justify-items en el contenedor de tu grid. Esta propiedad acepta los mismos valores aprendidos en los dos desafíos previos, siendo la única diferencia que moverá all los elementos de la grid hacia el alineamiento deseado.

- ejercicio de freecode-->Usa esta propiedad para centrar todos los elementos horizontalmente.

{Lección 13} Alinear todos los elementos verticalmente usando *align-items*:

Usar la propiedad align-items en el contenedor de una grid establecerá el alineamiento vertical de todos los elementos de la grid.

- ejercicio del freecode-->Úsala ahora para mover todos los elementos hacia el final de cada celda.

{Lección 14} Dividir la cuadrícula en una plantilla de área:

Puedes agrupar las celdas de tu grid en una área y darle a esa área un nombre personalizado. Haz esto usando grid-template-areas en el contenedor de la siguiente manera

grid-template-areas:

"header header header"

"advert content content"

"footer footer footer";

El código anterior fusiona las tres celdas superiores en una área llamada header, las tres celdas inferiores en una área footer y hace dos áreas en la fila del medio; advert y content.

Nota: cada palabra en el código representa una celda y cada par de comillas representa una fila. Además de los nombres personalizados, puedes usar un punto (.) para designar una celda vacía en la grid.

- ejercicio del freecode-->Acomoda la plantilla de área de manera que la celda llamada advert se convierta en una celda vacía.

{Lección 15} Ubicar elementos en áreas de cuadrícula usando la propiedad *grid-area*:

Después de crear una plantilla de área para tu contenedor de cuadrícula (grid), cómo se muestra en el desafío anterior, puedes ubicar un elemento en tu área personalizada referenciando el nombre que le diste. Para hacer esto, usa la propiedad grid-area sobre un elemento así

```
.item1 {
  grid-area: header;
}
```

Esto le dice a la cuadrícula que quieres que la clase item1 se ubique en el área llamada header. En este caso, el elemento usará la totalidad de la fila superior porque esa área se llama header.

- ejercicio del freecode-->Ubica un elemento con clase item5 en el área footer usando la propiedad grid-area.

{Lección 16} Usar *grid-area* sin crear plantillas de área:

La propiedad grid-area que aprendiste en el último desafío puede ser usada de otra manera. Si tu cuadrícula (grid) no tiene plantillas de área de referencia, puedes crear un área para un elemento sobre la marcha para ubicar los elementos de la siguiente manera:

item1 { grid-area: 1/1/2/4; }

Esto usa los números de líneas aprendidos anteriormente para definir cuál será el área de ese elemento. Los números en el ejemplo anterior representan estos valores

grid-area: horizontal line to start at / vertical line to start at / horizontal line to end at / vertical line to end at;

Entonces, el elemento en el ejemplo ocupará las filas entre las líneas 1 y 2, y las columnas entre las líneas 1 y 4.

- ejercicio del freecode-->Usa la propiedad grid-area para ubicar el elemento con clase item5 entre la tercera y cuarta línea horizontal y entre la primera y cuarta línea vertical.

{Lección 17} Reducir repeticiones usando la función *repeat*:

Cuando usaste grid-template-columns y grid-template-rows para definir la estructura de una grid, ingresaste un valor para cada fila o columna que creaste.

Digamos que quieres una grid con 100 filas del mismo tamaño. No es muy práctico insertar 100 valores manualmente. Afortunadamente, hay una mejor manera - usando la función repeat para especificar el número de veces que quieres que tu columna o fila se repita, seguido de una coma y el valor que quieres repetir.

A continuación un ejemplo que crearía una cuadrícula de 100 filas, cada fila con 50px de alto.

grid-template-rows: repeat(100, 50px);

También puedes repetir múltiples valores con la función *repeat* e insertar la función entre otros valores al definir una estructura de grid. Así se ve

grid-template-columns: repeat(2, 1fr 50px) 20px;

Esto traduce a

grid-template-columns: 1fr 50px 1fr 50px 20px;

Nota: el 1fr 50px es repetido dos veces, seguido de 20px.

- ejercicio del freecode-->Usa repeat para eliminar repetición de la propiedad grid-template-columns.

{Lección 18} Limitar el tamaño del elemento usando la función *minmax*:

Hay otra función integrada para usar con grid-template-columns y grid-template-rows llamada minmax. Se usa para limitar el tamaño de los elementos cuando el contenedor de la cuadrícula (grid) cambia de tamaño. Para hacer esto, necesitas especificar el rango de tamaño aceptable para tu elemento. A continuación un ejemplo

grid-template-columns: 100px minmax(50px, 200px);

En el código anterior, grid-template-columns se configuró para crear dos columnas; la primera tiene 100px de ancho y la segunda tiene un ancho mínimo de 50px y máximo de 200px.

- ejercicio del freecode-->Usando la función minmax, reemplaza 1fr en la función repeat con un tamaño de columna que tenga como mínimo de ancho 90px y máximo de 1fr, y cambia el tamaño del panel de la vista previa para ver el efecto.

{Lección 19} Crear diseños flexibles usando *auto-fill*:

La función de repetición viene con una opción llamada auto-fill. Esto te permite insertar automáticamente tantas filas o columnas del tamaño deseado como sea posible, dependiendo del tamaño del contenedor. Puedes crear diseños flexibles al combinar auto-fill con minmax, así

repeat(auto-fill, minmax(60px, 1fr));

Cuando el tamaño del contenedor cambia, esta configuración sigue insertando columnas de 60px y estirándolas hasta que pueda insertar otra.

Nota: Si a tu contenedor no le caben todos los elementos en una fila, los moverá hacia abajo a una nueva fila.

- ejercicio del freecode-->En la primera cuadrícula (grid), usa auto-fill con repeat para rellenar la cuadrícula con columnas que tengan un ancho mínimo de 60px y máximo de 1fr. Luego, cambia el tamaño de la vista previa para ver auto-fill en acción.

{Lección 20} Crear diseños flexibles usando *auto-fit*:

auto-fit funciona casi igual que auto-fill. La única diferencia es que cuando el tamaño del contenedor excede el tamaño de todos sus elementos combinados, auto-fill sigue insertando filas o columnas vacías y empuja los elementos hacia un lado, mientras que auto-fit colapsa esas filas o columnas y estira los elementos para que cubran el tamaño del contenedor.

Nota: si a tu contenedor no le caben todos los elementos en una fila, los moverá hacia abajo a una nueva fila.

- ejercicio del freecode-->En la segunda cuadrícula (grid), usa auto-fit con repeat para rellenar la cuadrícula con columnas que tengan un ancho mínimo de 60px y máximo de 1fr. Luego cambia el tamaño de la vista previa para ver la diferencia.

{Lección 21} Usar consultas de medios (media queries) para crear diseños responsivos:

CSS Grid puede ser una manera fácil de hacer que tu sitio sea más receptivo al usar consultas de medios (media queries) para reorganizar las áreas de cuadrícula (grid), cambiar sus dimensiones y reorganizar la ubicación de los elementos.

En la vista previa, cuando el ancho del viewport es 300px o más, el número de columnas cambia de 1 a 2. El área de publicidad entonces ocupa la columna de la izquierda por completo.

- ejercicio del freecode--> Cuando el ancho del viewport sea 400px o más, haz que el área header ocupe la fila superior por completo y que el área footer ocupe la fila inferior por completo.

{Lección 22} Crear cuadrículas (grids) dentro de cuadrículas:

Convertir un elemento a una cuadrícula solo afecta el comportamiento de sus descendientes directos. Entonces, al convertir un descendiente directo a una cuadrícula, obtienes una cuadrícula dentro de una cuadrícula.

Por ejemplo, al establecer las propiedades display and grid-template-columns del elemento con clase item3, creas una cuadrícula dentro de tu cuadrícula.

- ejercicio del freecode-->Convierte el elemento con clase item3 en una cuadrícula con dos columnas de ancho auto y 1fr usando display y grid-template-columns.