

Unidad 3 – Parte 2

Hojas de estilo en cascada: CSS y CSS3

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN
CFGs DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Índice

1. Introducción
2. CSS Flexbox
3. CSS Grid Layout
4. Propiedades de CSS avanzadas
 1. Cursor del ratón
 2. Bordes
 3. Degradados
 4. Sombra
 5. Efectos de texto
 6. Fuentes
 7. Transformaciones
 8. Transiciones
 9. Animaciones
 10. Columnas
 11. Interfaz de usuario
 12. Box Sizing
 13. Media Queries

Índice

1. **Introducción**
2. CSS Flexbox
3. CSS Grid Layout
4. Propiedades de CSS avanzadas
 1. Cursor del ratón
 2. Bordes
 3. Degradados
 4. Sombra
 5. Efectos de texto
 6. Fuentes
 7. Transformaciones
 8. Transiciones
 9. Animaciones
 10. Columnas
 11. Interfaz de usuario
 12. Box Sizing
 13. Media Queries

1. Introducción

- ✓ **Objetivo inicial de CSS:** separar el contenido de la forma.
- ✓ **CSS 3:** Incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas y mayor número de efectos visuales, sin tener que recurrir a trucos, que a menudo complicaban el código de las web.
- ✓ Código más simple para muchas tareas.
- ✓ Mayores opciones de gráficas.



- ✓ **Desventaja:** No tiene compatibilidad 100% con ningún navegador. Para determinadas propiedades debemos usar los prefijos -moz- y -webkit- para que funcionen en navegadores basados en motores Gecko y WebKit, como Firefox, Safari y Google Chrome.

Índice

1. Introducción
2. **CSS Flexbox**
3. CSS Grid Layout
4. Propiedades de CSS avanzadas
 1. Cursor del ratón
 2. Bordes
 3. Degradados
 4. Sombra
 5. Efectos de texto
 6. Fuentes
 7. Transformaciones
 8. Transiciones
 9. Animaciones
 10. Columnas
 11. Interfaz de usuario
 12. Box Sizing
 13. Media Queries

2. CSS Flexbox

- ✓ El Módulo de Caja Flexible, comúnmente llamado *flexbox*, fue diseñado como un modelo unidimensional de layout, y como un método que pueda ayudar a distribuir el espacio entre los ítems de una interfaz y mejorar las capacidades de alineación.
- ✓ Flexbox provee herramientas para crear layouts complejos y flexibles que, históricamente, han sido difíciles de implementar en CSS.
- ✓ Permite colocar los elementos de una página para que se comporten de forma predecible cuando el diseño de la página debe acomodarse a diferentes tamaños de pantalla y diferentes dispositivos. Para muchas aplicaciones, el modelo "caja flexible" produce una mejora sobre el modelo "bloque" porque no utiliza la propiedad float, ni hace que los márgenes del contenedor flexible interfieran con los márgenes de sus contenidos.
- ✓ https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout

2. CSS Flexbox

- ✓ La principal idea que hay detrás de la maquetación **FLEX** es que vamos a tener un elemento, es decir, una etiqueta que va a poder *controlar* las propiedades de los elementos que contiene.
- ✓ Por lo tanto, en esta situación, vamos a poder distinguir dos tipos de elementos:
 - El **contenedor flex** que tendrá asignada la propiedad CSS ***display:flex*** y que va a controlar ciertas propiedades de los elementos que contiene. Añadiendo esta propiedad podremos ver que de manera automática los elementos ajustan su anchura a su contenido y flotan a la izquierda.
 - Los **elementos flexibles** que son los elementos que están dentro del contenedor y cuyas propiedades modificaremos.
- ✓ Podremos modificar propiedades CSS como:
 - La altura y anchura de los elementos flexibles.
 - El orden en el que se nos van a presentar.
 - La alineación vertical y horizontal.
 - La distribución de los elementos flexibles a lo largo del contenedor.
- ✓ Es decir, vamos a poder controlar propiedades que usamos para maquetar y además, vamos a poder maquetar de manera mucho más ágil a lo que lo hacemos con las técnicas tradicionales de maquetado.

2. CSS Flexbox

✓ Propiedades en el contenedor Flex

✓ Dirección de los elementos flexibles:

- Propiedad **flex-direction**: Puede tomar los siguientes valores:
 - **row**: es la opción por defecto y ajustará los elementos flexibles de izquierda a derecha.
 - **row-reverse**: igual que la anterior pero de derecha a izquierda.
 - **column**: ajustará los elementos flexibles en columna, de arriba a abajo.
 - **column-reverse**: igual que la de arriba pero de abajo a arriba.

2. CSS Flexbox

✓ Propiedades en el contenedor Flex

✓ Ajuste de los elementos flexibles:

- Por defecto y sin indicar ninguna anchura, los elementos flexibles adecuan su tamaño a su contenido y se colocan todos a la izquierda permaneciendo siempre así aunque el ancho de la pantalla sea muy pequeño. Esto puede provocar desajustes en pantallas muy pequeñas.
- Propiedad **flex-wrap**: Puede tomar los siguientes valores:
 - **no-wrap**: es el valor por defecto y fuerza para que siempre los elementos estén en la misma línea aunque esto suponga que se salgan del contenedor (les haya dado o no les haya dado anchura).
 - **wrap**: provoca un salto de línea si la anchura de los elementos (fijada por nosotros o por el contenedor) es superior a la del contenedor.
 - **wrap-reverse**: lo mismo que arriba pero de abajo a arriba.
- Las dos propiedades, **flex-direction** y **flex-wrap** podemos juntarlas en la propiedad **flex-flow** con dos partes como por ejemplo:

```
flex-flow: row wrap;
```

2. CSS Flexbox

✓ Propiedades en el contenedor Flex

✓ Alineación horizontal de los elementos flexibles:

- Podemos alinear horizontalmente los elementos flexibles, tengan o no tengan establecida una anchura.
- Propiedad **justify-content**: Puede tomar los siguientes valores:
 - **flex-start**: Los elementos flexibles se sitúan al principio.
 - **flex-end**: Los elementos flexibles se sitúan al final.
 - **center**: Los elementos se centran horizontalmente.
 - **space-between**: Distribuye el espacio restante entre los elementos pero el primero y el último están en los bordes.
 - **space-around**: Distribuye el espacio restante entre los elementos pero no tiene en cuenta la distancia a los bordes.
 - **space-evenly**: Distribuye el espacio restante entre los elementos y tiene en cuenta la distancia a los bordes.

2. CSS Flexbox

✓ Propiedades en el contenedor Flex

✓ Alineación vertical de los elementos flexibles:

- Propiedad **align-items**: Puede tomar los siguientes valores:
 - **flex-start**: Los elementos se ponen junto al borde superior.
 - **flex-end**: Los elementos se ponen junto al borde inferior.
 - **center**: Los elementos flexibles se centran verticalmente.
 - **stretch**: Los elementos crecen en altura para ocupar toda la altura del contenedor flexible. No deben tener altura fija establecida.
 - **baseline**: Los elementos se alinean en relación con la primera línea de texto que posean los elementos flexibles.

2. CSS Flexbox

✓ Propiedades en el contenedor Flex

✓ Alineación vertical – Wrap (cuando tengo varias líneas):

- Si hemos usado la propiedad ***flex-wrap:wrap*** y resulta que tenemos varias líneas de elementos flexibles también se pueden alinear usando la propiedad CSS **align-content**. Los valores que puede tomar son análogos a los vistos anteriormente:
 - **flex-start**
 - **flex-end**
 - **center**
 - **stretch**
 - **space-between**
 - **space-around**

2. CSS Flexbox

✓ Elementos flexibles:

- ✓ Además de modificando las propiedades del contenedor FLEX, podemos actuar sobre los elementos flexibles modificando otra serie de propiedades relacionadas.

✓ Orden de los elementos flexibles:

- Los elementos flexibles se muestran dentro del contenedor flex en el mismo orden en que están escritos en nuestro código HTML.
- Si queremos modificar esto debemos añadir la propiedad CSS ***order*** a los **elementos** cuyo orden queremos modificar.
- Por defecto este valor es 0 y se mostrarán primeros aquellos elementos que tenga un mayor orden. En caso de empate se muestra antes el que primero estuviera en el código.

2. CSS Flexbox

✓ Elementos flexibles:

✓ Tamaño de los elementos flexibles:

- **flex-grow:** que sirve para indicar, mediante un número, el factor de crecimiento de un elemento flexible cuando se distribuye entre los elementos flexibles el espacio restante. Por defecto es 1 pero si queremos que un elemento participe en el reparto es necesario añadir esta propiedad.
- **flex-shrink:** que sirve para indicar, mediante un número el factor de contracción de un elemento flexible cuando el tamaño de todos sobrepasa el tamaño del contenedor. Por defecto es 1 pero si quiero que un elemento participe en la contracción es necesario añadir esta propiedad.
- **flex-basis:** que sirve para indicar el tamaño de un elemento antes de que el espacio restante (negativo positivo) se distribuya. Por defecto el valor de esta propiedad es *auto* y hace que la anchura del elemento flexible se ajusta a su contenido.

2. CSS Flexbox

✓ Elementos flexibles:

- ✓ En ocasiones podemos necesitar que un elemento flexible tenga una alineación vertical diferente al resto. En este caso, en el elemento para el que quiero una alineación diferente, debo añadir la propiedad CSS **align-self** que puede tomar los mismo valores (y con el mismo significado) que la propiedad **align-items**.
 - **flex-start**
 - **flex-end**
 - **center**
 - **stretch** (no debe tener altura establecida)
 - **baseline**
- ✓ **NOTA:** Los elementos flex no hace caso a la propiedad *vertical-align*.

Índice

1. Introducción
2. CSS Flexbox
3. **CSS Grid Layout**
4. Propiedades de CSS avanzadas
 1. Cursor del ratón
 2. Bordes
 3. Degradados
 4. Sombra
 5. Efectos de texto
 6. Fuentes
 7. Transformaciones
 8. Transiciones
 9. Animaciones
 10. Columnas
 11. Interfaz de usuario
 12. Box Sizing
 13. Media Queries

3. CSS Grid Layout

- ✓ La idea principal que hay detrás de la maquetación **GRID** es que vamos a tener un elemento, es decir, una etiqueta que nos va a permitir *controlar* propiedades de los elementos que contiene y establecer **estructuras complejas** para distribuirlos.
- ✓ https://developer.mozilla.org/es/docs/Web/CSS/CSS_Grid_Layout
- ✓ Por lo tanto, en esta situación, vamos a poder distinguir dos tipos de elementos:
 - El **contenedor GRID** que tendrá asignada la propiedad CSS *display:grid* (si queremos que nuestra rejilla sea un elemento de bloque) o *display:inline-grid* (si queremos que nuestra rejilla sea un elemento en línea). Todos los elementos que contiene pasan a convertirse de manera automática en elementos del GRID cuya colocación y propiedades podremos empezar a modificar desde el contenedor.
 - Los **elementos GRID** que son los elementos que están dentro del contenedor, elementos que distribuiremos y cuyas propiedades modificaremos.
- ✓ Desde el contenedor podemos modificar:
 - La estructura en filas y columnas y la separación entre ellas.
 - Definir áreas del GRID con nombre.
 - La alineación horizontal y vertical de los elementos del GRID y del propio GRID dentro del elemento que lo contiene.

3. CSS Grid Layout

✓ Propiedades en el contenedor Grid

✓ Definición de la estructura del GRID:

- **grid-template-columns:** Número de columnas del grid y tamaño de las mismas.
- **grid-template-rows:** Número de filas del grid y tamaño de las mismas.
- **grid-row-gap:** Para establecer la separación entre las diferentes filas.
- **grid-column-gap:** Para establecer la separación entre las diferentes columnas.

3. CSS Grid Layout

- ✓ Propiedades en el contenedor Grid
 - ✓ Definición de la estructura del GRID:
 - Ejemplo con columnas:

```
/* Tres columnas que se reparten el 100% del contenedor*/
```

```
grid-template-columns: 20% 50% 30%;
```

```
/* Cuatro columnas. Tres de tamaño fijo 100px y la otra ocupa el espacio libre restante */
```

```
grid-template-columns: 100px auto 100px 100px;
```

```
/* Cuatro columnas. Todas con un tamaño igual */
```

```
grid-template-columns: auto auto auto auto;
```

```
/* Tres columnas cada una con nombre (entre []). Dos con tamaño fijo y la otra ocupando el espacio restante */
```

```
grid-template-columns: [id] 100px [nombre] 300px [apellidos] auto;
```

3. CSS Grid Layout

- ✓ Propiedades en el contenedor Grid
 - ✓ Definición de la estructura del GRID:
 - Ejemplo con filas:

```
/* Tres filas que se reparten toda la altura del contenedor (la que sea) */  
grid-template-rows: 20% 50% 30%;
```

```
/* Cuatro filas. Tres de altura fija 100px y la otra ocupará el resto del espacio libre hasta llenar todo el contenedor en altura.*/  
grid-template-rows: 100px auto 100px 100px;
```

```
/* Cuatro filas que se reparten de manera equitativa el alto del contenedor */  
grid-template-rows auto auto auto auto;
```

```
/* Tres filas (todas con nombre, entre corchetes) Dos de ellas con tamaño fijo y la restante ocupará todo el alto libre. */  
grid-template-rows: [uno] 100px [dos] 300px [tres] auto;
```

3. CSS Grid Layout

✓ Propiedades en el contenedor Grid

✓ Definición de la estructura del GRID:

- Se pueden repetir valores y usar la unidad *fr* para establecer ratios para que los elementos se repartan el espacio restante.

```
/* Cuatro columnas. Tres de 20% con nombre col-start. Y la último que ocupará el resto del espacio libre */  
grid-template-columns: repeat(3, 20% [col-start]) auto;  
/* Cuatro columnas. Una de tamaño fijo y las demás se reparten el espacio libre en 5 partes de la siguiente manera (2+1+2) */  
grid-template-columns: 2fr 100px 1fr 2rf;
```

3. CSS Grid Layout

- ✓ Propiedades en el contenedor Grid
 - ✓ Definición de la estructura del GRID:
 - Ejemplo:

```
<div class="container">
  <div>Primero</div>
  <div>Segundo</div>
  <div>Tercero</div>
  <div>Cuarto</div>
  <div>Quinto</div>
  <div>Sexto</div>
  <div>Séptimo</div>
</div>
```

```
.container {
  background-color: #aaa;
  display: grid;
  grid-column-gap: 10px;
  grid-row-gap: 20px;
  grid-template-columns: 20% auto 20%;
  grid-template-rows: repeat(3, 100px);
  margin: 20px auto;
  padding: 1em;
  width: 80%;
}

.container > div {
  background-color: bisque;
  border: 1px solid black;
}
```

3. CSS Grid Layout

- ✓ Propiedades en el contenedor Grid
 - ✓ Definición de la estructura del GRID:
 - Ejemplo:



3. CSS Grid Layout

✓ Propiedades en el contenedor Grid

✓ Alineación Horizontal:

- Por defecto los elementos del GRID ocupan todo el ancho de la celda que le corresponde pero podemos optar por otro tipo de alineaciones horizontales dando valores a la propiedad ***justify-items***. Los diferentes valores que puede tomar son los siguientes:
 - **start**
 - **end**
 - **center**
 - **stretch** (opción por defecto)

3. CSS Grid Layout

✓ Propiedades en el contenedor Grid

✓ Alineación Vertical:

- Por defecto los elementos del GRID ocupan todo el alto de la celda que le corresponde pero podemos optar por otro tipo de alineaciones verticales dando valores a la propiedad ***align-items***. Los diferentes valores que puede tomar son los siguientes:

- **start**
- **end**
- **center**
- **stretch** (opción por defecto)

- ✓ Se puede juntar las dos alineaciones (horizontal y vertical)usando la propiedad ***place-items*** indicando primero el valor para ***align-items*** (vertical) y después el valor para ***justify-items*** (horizontal):

```
place-items: start end;
```

3. CSS Grid Layout

✓ Propiedades en el contenedor Grid

✓ Distribución dentro del contenedor:

- En determinados casos puede suceder que los elementos del GRID no ocupen todo el ancho o todo el alto del contenedor GRID. En estas ocasiones puedo distribuir las columnas y las filas usando las propiedades **justify-content** (horizontal) y **align-content** (vertical).
- Se puede juntar las dos alineaciones (horizontal y vertical)usando la propiedad **place-content** indicando primero el valor para **align-content** (vertical) y después el valor para **justify-content** (horizontal):

```
place-content: start end;
```

3. CSS Grid Layout

✓ Elementos del Grid

- Los **Elementos GRID** son los hijos directos, dentro del árbol DOM de nuestra página HTML, del elemento con la propiedad CSS ***display:grid***.
- De manera individual podemos modificar las propiedades de estos elementos para conseguir lo siguiente:
 - Definir el área o zona del GRID (rejilla) que va a ocupar.
 - Especificar la alineación horizontal del elemento.
 - Especificar la alineación vertical del elemento.

3. CSS Grid Layout

✓ Elementos del Grid

✓ Área del elemento GRID:

- **grid-column-start:** especifica cuál es la columna de inicio.
- **grid-column-end:** especifica cuál es la columna de fin.
- **grid-row-start:** especifica cuál es la fila de inicio.
- **grid-row-end:** especifica cuál es la fila de fin.

- Podemos juntar estas propiedades:

```
/* Para juntar las dos propiedades referentes a columnas */  
grid-column: start / end;
```

```
/* Para juntar las dos propiedades referentes a filas */  
grid-row: start / end;
```

```
/* Para junta las cuatro propiedades que nos permiten definir el área */  
grid: row-start column-start row-end column-end;
```

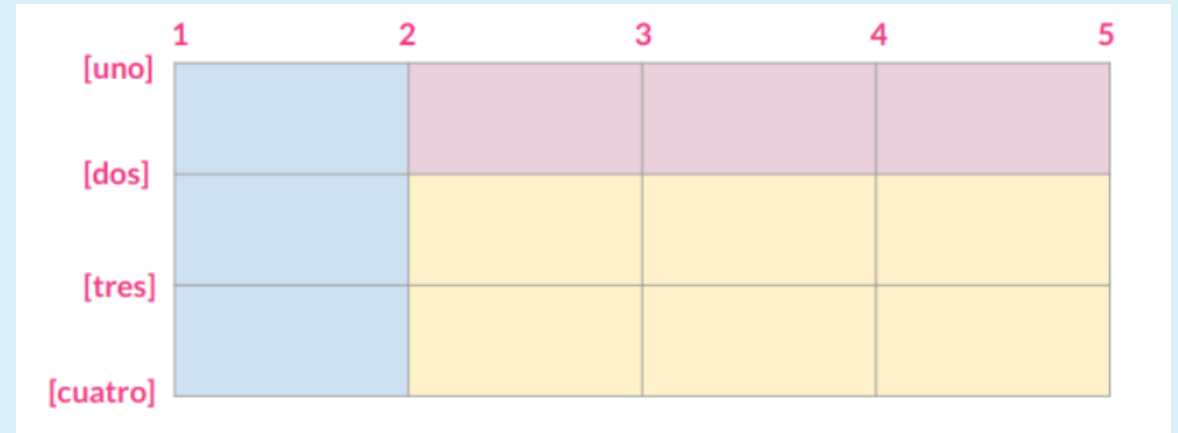
3. CSS Grid Layout

✓ Elementos del Grid

✓ Área del elemento GRID:

■ Ejemplo:

```
#azul {  
  background-color: blue;  
  grid-column-start: 1;  
  grid-column-end: 2;  
  grid-row-start: uno;  
  grid-row-end: cuatro;  
}  
  
#rojo {  
  background-color: red;  
  grid-column-start: 2;  
  grid-column-end: 5;  
  grid-row-start: uno;  
  grid-row-end: dos;  
}  
  
#amarillo {  
  background-color: yellow;  
  grid-column-start: 2;  
  grid-column-end: 5;  
  grid-row-start: dos;  
  grid-row-end: span 2;  
}
```



3. CSS Grid Layout

✓ Elementos del Grid

✓ Alineación horizontal:

- La alineación horizontal de un elemento de manera individual se consigue usando la propiedad ***justify-self*** que puede tomar los mismos valores y funciona igual que la propiedad *justify-items* que dábamos al contenedor GRID. Estos valores son:
 - start
 - end
 - center
 - stretch (por defecto)

3. CSS Grid Layout

✓ Elementos del Grid

✓ Alineación vertical:

- La alineación vertical de un elemento de manera individual se consigue usando la propiedad ***align-self*** que puede tomar los mismos valores y funciona igual que la propiedad *align-items* que dábamos al contenedor GRID. Estos valores son:
 - start
 - end
 - center
 - stretch (por defecto)

3. CSS Grid Layout

✓ Áreas Grid

- ✓ Podemos maquetar únicamente nombrando áreas. Para ello usaremos las siguiente propiedades:
 - **grid-area** en los elementos GRID
 - **grid-template-area** en el contenedor GRID.

```
.container {  
  grid-template-columns: repeat(5, 20%);  
  grid-template-rows: repeat(3, 100px);  
}
```

```
#cab {  
  background-color: blue;  
  grid-area: cab;  
}  
  
#pie {  
  background-color: green;  
  grid-area: pie;  
}  
  
#menu {  
  background-color: red;  
  grid-area: menu;  
}  
  
#principal {  
  background-color: yellow;  
  grid-area: main;  
}
```


3. CSS Grid Layout

✓ Áreas Grid

- ✓ Añadimos al contenedor la propiedad **grid-template-area** que define la estructura sin dar ninguna propiedad adicional a los elementos del Grid.
- ✓ Cada nombre representa el elemento que va a ocupar una celda. De esta manera, el contenido de cada fila va entre “” y:
 - La **primera fila** será ocupada totalmente por el elemento con *grid-area: cab*;
 - En la **segunda fila**, la primera celda para el elemento con *grid-area: menu*, luego un hueco que se indica con . y posteriormente tres celdas para el elemento con *grid-area: main*.
 - En la **tercera fila** la primera celda es para el elemento con *grid-area: menu* y el resto para el que tenga *grid-area: pie*

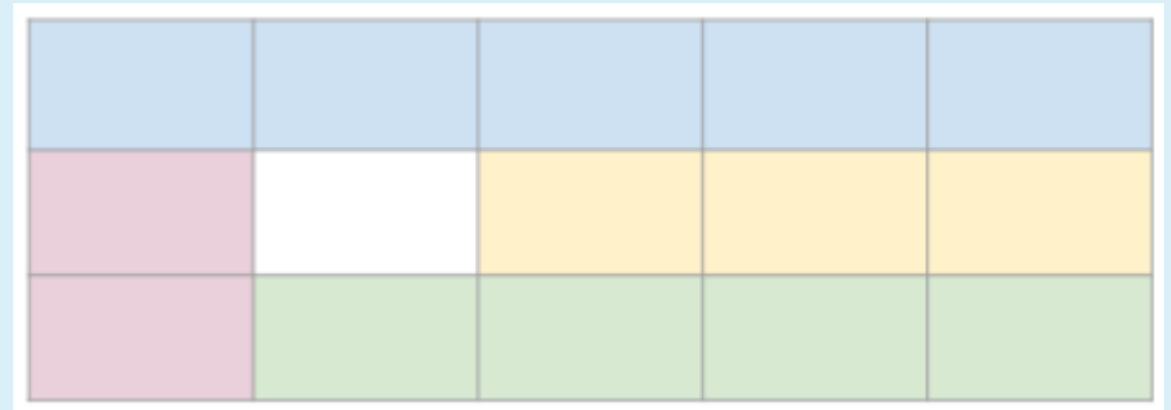
```
.container {  
  display: grid;  
  grid-template-columns: repeat(5, 20%);  
  grid-template-rows: repeat(3, 100px);  
  grid-template-areas:  
    "cab cab cab cab cab"  
    "menu . main main main"  
    "menu pie pie pie pie";  
}
```

3. CSS Grid Layout

✓ Áreas Grid

```
#cab {  
  background-color: blue;  
  grid-area: cab;  
}  
  
#pie {  
  background-color: green;  
  grid-area: pie;  
}  
  
#menu {  
  background-color: red;  
  grid-area: menu;  
}  
  
#principal {  
  background-color: yellow;  
  grid-area: main;  
}
```

```
.container {  
  display: grid;  
  grid-template-columns: repeat(5, 20%);  
  grid-template-rows: repeat(3, 100px);  
  grid-template-areas:  
    "cab cab cab cab cab"  
    "menu . main main main"  
    "menu pie pie pie pie";  
}
```



Índice

1. Introducción
2. CSS Flexbox
3. CSS Grid Layout
4. **Propiedades de CSS avanzadas**
 1. Cursor del ratón
 2. Bordes
 3. Degradados
 4. Sombra
 5. Efectos de texto
 6. Fuentes
 7. Transformaciones
 8. Transiciones
 9. Animaciones
 10. Columnas
 11. Interfaz de usuario
 12. Box Sizing
 13. Media Queries

4. Propiedades de CSS avanzadas

4.1. Propiedad para cambiar el tipo de cursor del ratón

- ✓ Especifica el cursor del ratón que se mostrará al apuntar sobre un elemento.
- ✓ Sintaxis: *cursor: value;*
- ✓ https://www.w3schools.com/cssref/pr_class_cursor.asp

4. Propiedades de CSS avanzadas

4.2. Propiedades de bordes

- ✓ **border-radius:** crea las esquinas redondeadas, especificando las medidas del radio que deben darse a la curva de las esquinas.
 - https://www.w3schools.com/cssref/css3_pr_border-radius.asp
- ✓ **border-imagen:** utilizar imágenes como bordes. Sintaxis: `border-image: source slice width outset repeat|initial|inherit;`
 - *source*: Para indicar la URL de la imagen que vamos a utilizar como borde.
 - *slice*: Indica el espacio de la imagen que será visible como borde, en los cuatro lados del elemento, es decir, top, right, bottom y left.
 - *width*: Para indicar la anchura del borde.
 - *outset*: Nos sirve para indicar el área en la que la imagen de borde se extiende más allá del área del elemento, en los 4 lados del mismo.
 - *repeat*: Permite marcar si se desea o no que se repita la imagen del borde haciendo un mosaico o si se desea que se estire, etc.
 - https://www.w3schools.com/cssref/css3_pr_border-image.asp

4. Propiedades de CSS avanzadas

4.3. Propiedades para definir un gradiente

- ✓ Hasta la llegada de CSS 3, necesitábamos usar imágenes como fondo de los elementos para implementarlos.
- ✓ Implementan un gradiente de color, que pasa de un estado a otro a lo largo del fondo de los elementos HTML, ya sea capas, elementos de listas, botones, etc.
- ✓ Se obtendrán por medio de la especificación de una serie de características, como la posición inicial, la dirección hacia donde se realizará, si es circular o linear, y los colores que se incorporarán en cada uno de los pasos del gradiente.
- ✓ El navegador es el encargado de renderizar el gradiente en función de las características escritas para definirlo. Podemos asignar gradientes como fondo en cualquier elemento HTML donde se podía implementar una imagen de fondo.
- ✓ Tipos:
 - Gradientes lineales (va abajo / arriba / izquierda / derecha / diagonal)
 - Gradientes radiales (definido por su centro)
- ✓ https://www.w3schools.com/css/css3_gradients.asp

4. Propiedades de CSS avanzadas

4.3. Propiedades para definir un gradiente

✓ Gradientes lineales:

- Se crea un gradiente que va de un color a otro de manera lineal. Puede ser de arriba a abajo, de izquierda a derecha y viceversa. Incluso se puede conseguir un degradado en un gradiente de una línea con cualquier ángulo.
- Sintaxis: `background-image: linear-gradient(direction, color-stop1, color-stop2, ...);`

4. Propiedades de CSS avanzadas

4.3. Propiedades para definir un gradiente

✓ Gradientes lineales:

- Ejemplos:
 - Gradiente lineal de izquierda a derecha:

```
#grad1 {  
  height: 200px;  
  background-color: red; /* For browsers that do not support gradients */  
  background-image: linear-gradient(to right, red , yellow); /* Standard syntax (must be last) */  
}
```



4. Propiedades de CSS avanzadas

4.3. Propiedades para definir un gradiente

✓ Gradientes lineales:

- Ejemplos:
 - Gradiente lineal que comienza en la parte superior izquierda (y va hacia la parte inferior derecha). Empieza en rojo, pasando a amarillo:

```
#grad1 {  
  height: 200px;  
  background-color: red; /* For browsers that do not support gradients */  
  background-image: linear-gradient(to bottom right, red, yellow); /* Standard syntax (must be last) */  
}
```



4. Propiedades de CSS avanzadas

4.3. Propiedades para definir un gradiente

✓ Gradientes lineales:

■ Ejemplos:

- Usando transparencia: e puede usar para crear efectos de desvanecimiento. Se utiliza la función rgba (). El último parámetro de la función rgba () puede ser un valor de 0 a 1, y se define la transparencia del color: 0 indica una transparencia total, 1 indica todo color (sin transparencia).

```
#grad1 {  
  height: 200px;  
  background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1)); /* Standard syntax (must be last)  
  */  
}
```



4. Propiedades de CSS avanzadas

4.3. Propiedades para definir un gradiente

✓ Gradientes radiales:

- Se implementa un gradiente que se distribuye radialmente, desde un punto del elemento hacia fuera, de manera circular, que puede tener el mismo valor de radio (para hacer degradados en círculos perfectos) o con valores de radio variables (lo que generaría elipses).
- Sintaxis: `background-image: radial-gradient(shape size at position, start-color, ..., last-color);`
 - Posición inicial del gradiente circular: se especifica con una o dos coordenadas, que pueden tener distintas unidades CSS. Si se omite, se entiende que el degradado tiene que comenzar en el punto central del fondo del elemento
 - La forma puede ser circular o elipse, para lo cual especificamos las palabras "*circle*" o "*ellipse*". El tamaño lo expresamos con otra serie de palabras clave, que indican hasta donde debe crecer el círculo o elipse: *closest-side* | *closest-corner* | *farthest-side* | *farthest-corner*. Por ejemplo, *closest-side* indica que el círculo o elipse debe crecer hasta el lado más cercano. La palabra *farthest-corner* indicaría que debe crecer hasta la esquina más lejana.

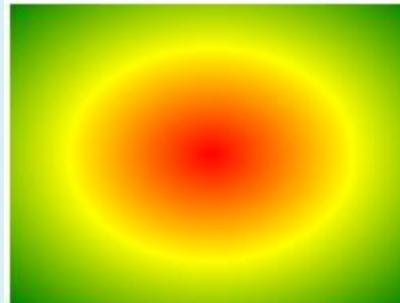
4. Propiedades de CSS avanzadas

4.3. Propiedades para definir un gradiente

✓ Gradientes radiales:

- Ejemplos:
 - Gradiente radial con forma de elipse (por defecto):

```
#grad1 {  
  height: 150px;  
  width: 200px;  
  background-color: red; /* For browsers that do not support gradients */  
  background-image: radial-gradient(red, yellow, green); /* Standard syntax (must be last) */  
}
```



4. Propiedades de CSS avanzadas

4.3. Propiedades para definir un gradiente

✓ Gradientes radiales:

- Ejemplos:
 - Gradiente radial con forma de círculo:

```
#grad2 {  
  height: 150px;  
  width: 200px;  
  background-color: red; /* For browsers that do not support gradients */  
  background-image: radial-gradient(circle, red, yellow, green); /* Standard syntax (must be last) */  
}
```



4. Propiedades de CSS avanzadas

4.4. Propiedades para dar efectos de sombra:

- ✓ **text-shadow:** Aplica sombra al texto. Se especifica la sombra horizontal, la sombra vertical y el color de la sombra.

```
h1 {  
  text-shadow: 2px 2px red;  
}
```

Text-shadow effect!

- ✓ **box-shadow:** añadir sombras a las cajas.

```
div {  
  width: 300px;  
  height: 100px;  
  padding: 15px;  
  background-color: yellow;  
  box-shadow: 10px 10px grey;  
}
```

This is a div element with a box-shadow

- ✓ https://www.w3schools.com/css/css3_shadows.asp

4. Propiedades de CSS avanzadas

4.5. Propiedades para dar efectos al texto:

- ✓ **word-wrap:** sirve para especificar que las palabras que sean demasiado largas se deben cortar, de manera que quepan en el ancho disponible de la caja.

```
p.test {  
  width: 11em;  
  border: 1px solid #000000;  
  word-wrap: break-word;  
}
```

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

- ✓ **text-overflow:** especifica cómo se debe señalar al usuario el contenido desbordado que no se muestra. Se puede recortar, mostrar puntos suspensivos (...) o mostrar una cadena personalizada. Las dos propiedades siguientes son necesarias: *white-space: nowrap; overflow: hidden.*

4. Propiedades de CSS avanzadas

4.5. Propiedades para dar efectos al texto:

- ✓ **text-overflow:** especifica cómo se debe señalar al usuario el contenido desbordado que no se muestra. Se puede recortar, mostrar puntos suspensivos (...) o mostrar una cadena personalizada. Las dos propiedades siguientes son necesarias: *white-space: nowrap; overflow: hidden.*

```
p.test1 {  
  white-space: nowrap;  
  width: 200px;  
  border: 1px solid #000000;  
  overflow: hidden;  
  text-overflow: clip;  
}  
  
p.test2 {  
  white-space: nowrap;  
  width: 200px;  
  border: 1px solid #000000;  
  overflow: hidden;  
  text-overflow: ellipsis;  
}
```

text-overflow: clip:

This is some long text that will

text-overflow: ellipsis:

This is some long text that ...

4. Propiedades de CSS avanzadas

4.5. Propiedades para dar efectos al texto:

- ✓ **word-break:** especifica cómo deben romperse las palabras al llegar al final de una línea.

```
p.test1 {  
  width: 140px;  
  border: 1px solid #000000;  
  word-break: keep-all;  
}
```

```
p.test2 {  
  width: 140px;  
  border: 1px solid #000000;  
  word-break: break-all;  
}
```

This paragraph
contains some text.
This line will-break-
at-hyphens.

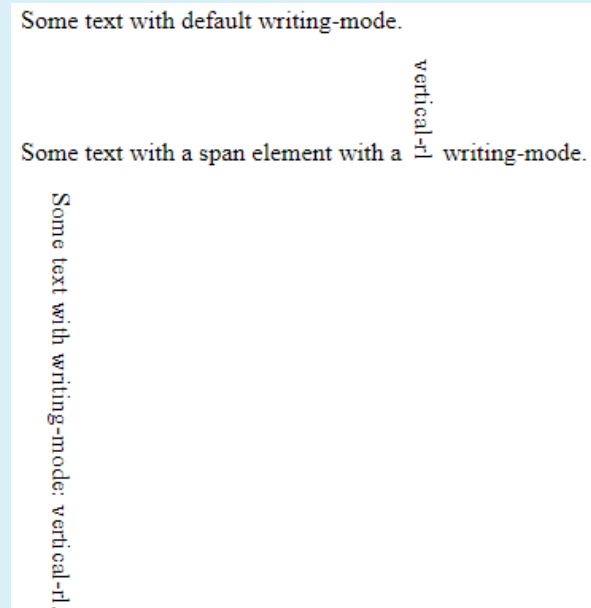
This paragraph contain
s some text. The lin
es will break at any c
haracter.

4. Propiedades de CSS avanzadas

4.5. Propiedades para dar efectos al texto:

- ✓ **writing-mode:** especifica si las líneas de texto se disponen horizontal o verticalmente.

```
p.test1 {  
  writing-mode: horizontal-tb;  
}  
  
span.test2 {  
  writing-mode: vertical-rl;  
}  
  
p.test2 {  
  writing-mode: vertical-rl;  
}
```



Some text with default writing-mode.

Some text with a span element with a vertical-rl writing-mode.

Some text with writing-mode: vertical-rl.

- ✓ https://www.w3schools.com/css/css3_text_effects.asp

4. Propiedades de CSS avanzadas

4.6. Propiedades para fuentes:

- ✓ A la hora de escoger una fuente para usar en una página web, tradicionalmente se encontraba la limitación de que el usuario tuviera ese tipo de letra instalada en su ordenador, porque de no ser así, los textos se mostrarían con otras tipografías, distintas a las que habíamos elegido.
- ✓ **@font-face:** permite definir en nuestra hoja de estilos cualquier tipo de tipografía, independientemente de si el usuario dispone de ella o no y para ello lo único que debe preocuparnos es que este instalada en nuestro servidor. Admite formatos tipográficos: .eot (Embedded OpenType); .ttf (True Type Fonts); .otf (OpenType Fonts).
- ✓ https://www.w3schools.com/css/css3_fonts.asp

4. Propiedades de CSS avanzadas

4.6. Propiedades para fuentes:

```
@font-face{  
  font-family:<nombre_fuente>;  
  src: <source>[,<source>]*;  
  [font-weight:<weight>;  
  [font-style:<style>;  
}
```

```
@font-face {  
  font-family: Vivaldi;  
  font-style: normal;  
  font-weight: normal;  
  src: url(VIVALDI0.eot);  
}
```

```
H2{  
  font-family: "Vivaldi";  
}
```

4. Propiedades de CSS avanzadas

4.6. Propiedades para fuentes:

✓ OBLIGATORIOS:

- *font-family*: Nombre de la fuente
- *src*: Ubicación en el servidor.

✓ OPCIONALES:

- *font-stretch*: Define cómo se debe estirar la fuente. El valor predeterminado es "normal"
- *font-style*: Define cómo se debe diseñar la fuente. El valor predeterminado es "normal"
- *font-weight*: Define el ancho la fuente. El valor predeterminado es "normal"
- *unicode-range*: Define el rango de caracteres Unicode que admite la fuente.

4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 2D:

- Una transformación es un efecto que permite un cambio de elemento de forma, tamaño y posición.
- La propiedad *transform* puede operar cuatro transformaciones básicas:
 - *rotate* (rotar)
 - *scale* (escalar)
 - *skew* (inclinarse)
 - *translate* (trasladar o mover)
 - *matrix* (permiten rotar, escalar, mover e inclinar elementos)
- https://www.w3schools.com/css/css3_2dtransforms.asp

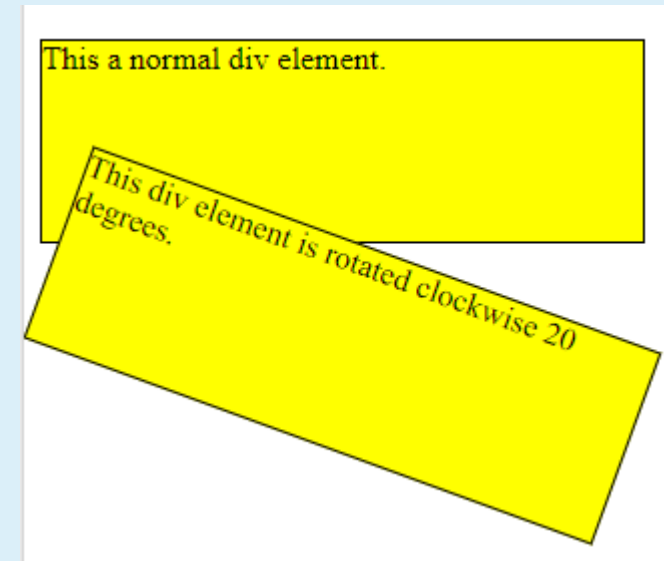
4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 2D:

- Método *rotate()*: Gira un elemento en sentido horario o antihorario según un grado dado. El uso de valores negativos girará el elemento en sentido contrario a las agujas del reloj.

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
div#myDiv {  
  -ms-transform: rotate(20deg); /* IE 9 */  
  -webkit-transform: rotate(20deg); /* Safari prior 9.0 */  
  transform: rotate(20deg); /* Standard syntax */  
}
```



4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 2D:

- Método *scale()*: Aumenta o disminuye el tamaño de un elemento (de acuerdo con los parámetros dados para el ancho y la altura).

```
div {  
  margin: 150px;  
  width: 200px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
  -ms-transform: scale(2,3); /* IE 9 */  
  -webkit-transform: scale(2,3); /* Safari prior 9.0 */  
  transform: scale(2,3); /* Standard syntax */  
}
```

This div element is two times
of its original width, and three
times of its original height.

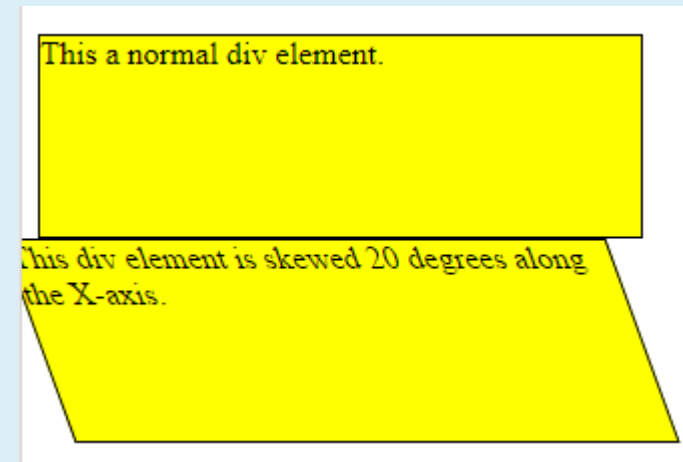
4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 2D:

- Método *skew()*: Sesga un elemento a lo largo de los ejes X e Y según los ángulos dados. Si no se especifica el segundo parámetro, tiene un valor cero.

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
div#myDiv {  
  -ms-transform: skewX(20deg); /* IE 9 */  
  -webkit-transform: skewX(20deg); /* Safari prior 9.0 */  
  transform: skewX(20deg); /* Standard syntax */  
}
```



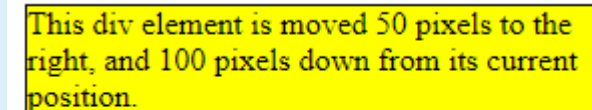
4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 2D:

- Método *translate()*: Mueve un elemento desde su posición actual (de acuerdo con los parámetros dados para el eje X y el eje Y).

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
  -ms-transform: translate(50px,100px); /* IE 9 */  
  -webkit-transform: translate(50px,100px); /* Safari prior 9.0 */  
  transform: translate(50px,100px); /* Standard syntax */  
}
```



This div element is moved 50 pixels to the right, and 100 pixels down from its current position.

4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 2D:

- Método *matrix()*: Combina todos los métodos de transformación 2D en uno. Toma seis parámetros, lo que le permite rotar, escalar, mover e inclinar elementos.
- *matrix (scaleX (), skewY (), skewX (), scaleY (), translateX (), translateY ())*

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
div#myDiv1 {  
  -ms-transform: matrix(1, -0.3, 0, 1, 0, 0); /* IE 9 */  
  -webkit-transform: matrix(1, -0.3, 0, 1, 0, 0); /* Safari prior 9.0 */  
  transform: matrix(1, -0.3, 0, 1, 0, 0); /* Standard syntax */  
}  
  
div#myDiv2 {  
  -ms-transform: matrix(1, 0, 0.5, 1, 150, 0); /* IE 9 */  
  -webkit-transform: matrix(1, 0, 0.5, 1, 150, 0); /* Safari prior 9.0 */  
  transform: matrix(1, 0, 0.5, 1, 150, 0); /* Standard syntax */  
}
```

This a normal div element.

Using the matrix() method.

Another use of the matrix() method.

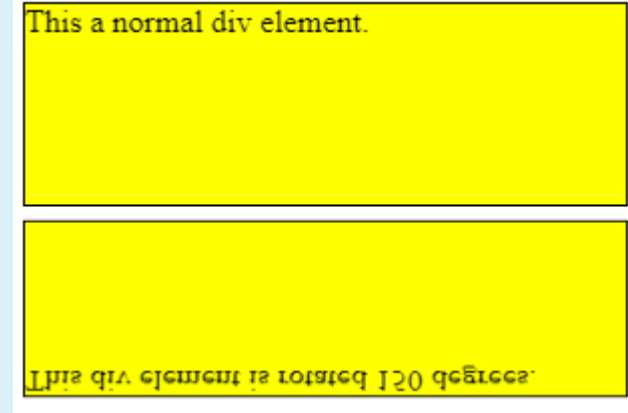
4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 3D:

- https://www.w3schools.com/css/css3_3dtransforms.asp
- Método *rotateX()*: Gira un elemento alrededor de su eje X en un grado dado.

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
#myDiv {  
  -webkit-transform: rotateX(150deg); /* Safari prior 9.0 */  
  transform: rotateX(150deg); /* Standard syntax */  
}
```



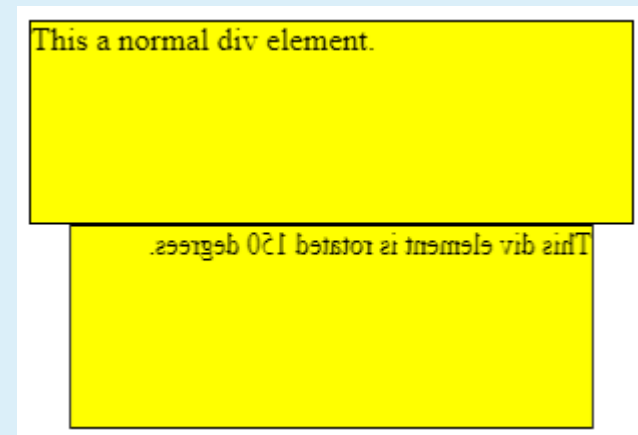
4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 3D:

- https://www.w3schools.com/css/css3_3dtransforms.asp
- Método *rotateY()*: Gira un elemento alrededor de su eje Y en un grado dado.

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
#myDiv {  
  -webkit-transform: rotateY(150deg); /* Safari prior 9.0 */  
  transform: rotateY(150deg); /* Standard syntax */  
}
```



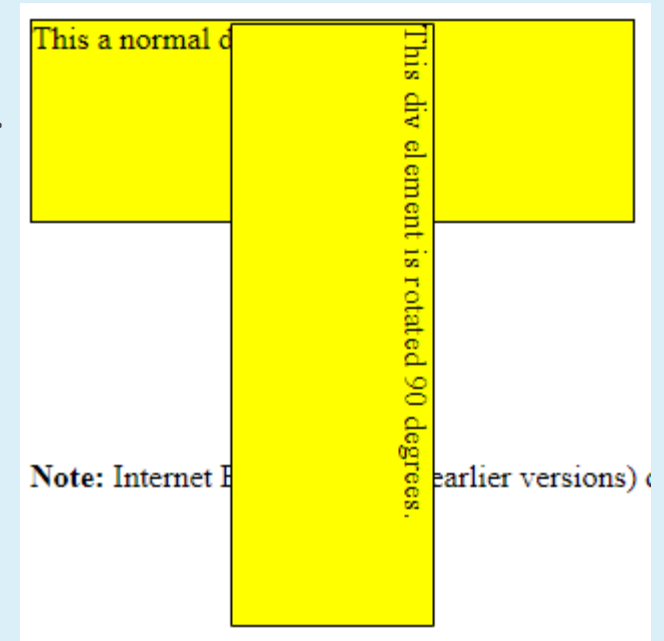
4. Propiedades de CSS avanzadas

4.7. Propiedades para transformaciones:

✓ Transformaciones 3D:

- https://www.w3schools.com/css/css3_3dtransforms.asp
- Método *rotateZ()*: Gira un elemento alrededor de su eje Z en un grado dado.

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
#myDiv {  
  -webkit-transform: rotateZ(90deg); /* Safari prior 9.0 */  
  transform: rotateZ(90deg); /* Standard syntax */  
}
```



4. Propiedades de CSS avanzadas

4.8. Propiedades para transiciones:

- ✓ Con CSS3, podemos agregar un efecto al cambiar de un estilo a otro, sin el uso de animaciones Flash o archivos JavaScript.
- ✓ **Transiciones:** Efectos que permiten un elemento cambiar gradualmente de un estilo a otro.
 - Para crear un efecto de transición, debe especificar dos cosas:
 - la propiedad CSS a la que desea agregar un efecto
 - la duración del efecto
 - **Nota:** Si la parte de la duración no se especifica, la transición no tendrá efecto, porque el valor predeterminado es 0.
- ✓ https://www.w3schools.com/css/css3_transitions.asp

4. Propiedades de CSS avanzadas

4.8. Propiedades para transiciones:

- ✓ **transition:** Una propiedad abreviada para configurar las cuatro propiedades de transición en una sola propiedad.
- ✓ **transition-delay:** Especifica un retraso (en segundos) para el efecto de transición.
- ✓ **transition-duration:** Especifica cuántos segundos o milisegundos tarda en completarse un efecto de transición.
- ✓ **transition-property:** Especifica el nombre de la propiedad CSS para la cual se aplica el efecto de transición.
- ✓ **transition-timing-function:** Especifica la curva de velocidad del efecto de transición.

4. Propiedades de CSS avanzadas

4.9. Propiedades para animaciones:

- ✓ Las animaciones CSS permiten la animación de la mayoría de los elementos HTML sin usar JavaScript o Flash.
- ✓ Una animación permite que un elemento cambie gradualmente de un estilo a otro.
- ✓ Puedes cambiar tantas propiedades CSS como quieras, tantas veces como quieras.
- ✓ Para usar la animación CSS, primero debe especificar algunos fotogramas clave para la animación.
- ✓ Los fotogramas clave contienen qué estilos tendrá el elemento en ciertos momentos.
- ✓ https://www.w3schools.com/css/css3_animations.asp

4. Propiedades de CSS avanzadas

4.9. Propiedades para animaciones:

✓ La regla @keyframes

- La regla @ keyframes es donde se crea la animación. Es necesario especificar un estilo CSS dentro de la regla @ keyframes y la animación cambiará gradualmente del estilo actual al nuevo estilo.
- Para que una animación funcione, se debe vincular la animación a un elemento.
- Ejemplo: Vincula la animación "ejemplo" al elemento <div>. La animación durará 4 segundos y cambiará gradualmente el color de fondo del elemento <div> de "rojo" a "amarillo".
- https://www.w3schools.com/css/tryit.asp?filename=trycss3_animation1

4. Propiedades de CSS avanzadas

4.9. Propiedades para animaciones:

- ✓ **@keyframes:** Especifica el código de animación.
- ✓ **animation:** Una propiedad abreviada para establecer todas las propiedades de animación.
- ✓ **animation-delay:** Especifica un retraso para el inicio de una animación.
- ✓ **animation-direction:** Especifica si una animación debe reproducirse hacia adelante, hacia atrás o en ciclos alternos.
- ✓ **animation:duration:** Especifica cuánto tiempo debe tomar una animación para completar un ciclo
- ✓ **animation-fill-mode:** Especifica un estilo para el elemento cuando la animación no se está reproduciendo (antes de que comience, después de que termine, o ambos)
- ✓ **animation-iteration-count:** Especifica el número de veces que se debe reproducir una animación.
- ✓ **animation-name:** Especifica el nombre de la animación @keyframes.
- ✓ **animation-play-state:** Especifica si la animación se está ejecutando o está en pausa.
- ✓ **animation-timing-function:** Especifica la curva de velocidad de la animación.

4. Propiedades de CSS avanzadas

4.9. Propiedades para animaciones:

- ✓ Ejemplo: Ejecuta una animación con todas las propiedades establecidas:
 - https://www.w3schools.com/css/tryit.asp?filename=trycss3_animation4
 - https://www.w3schools.com/css/tryit.asp?filename=trycss3_animation5 (Propiedad abreviada)

4. Propiedades de CSS avanzadas

4.10. Propiedades para diseñar varias columnas:

- ✓ El diseño de varias columnas de CSS permite una fácil definición de múltiples columnas de texto, como en los periódicos:
 - *column-count*: Especifica el número de columnas en que se debe dividir un elemento
 - *column-fill*: Especifica cómo rellenar columnas.
 - *column-gap*: Especifica el espacio entre las columnas.
 - *column-span*: Especifica cuántas columnas debe abarcar un elemento
 - *column-width*: Especifica un ancho óptimo sugerido para las columnas.
 - *columns*: Una propiedad abreviada para establecer el ancho de columna y el conteo de columnas
- ✓ https://www.w3schools.com/css/css3_multiple_columns.asp

4. Propiedades de CSS avanzadas

4.11. Propiedades para la interfaz de usuario:

- ✓ En CSS3, algunas de las nuevas características de la interfaz de usuario está cambiando el tamaño los elementos, caja de tamaño, y la esquematización.
- ✓ https://www.w3schools.com/css/css3_user_interface.asp
- ✓ La propiedad **resize** especifica si (y cómo) un elemento debe ser redimensionable por el usuario.
 - https://www.w3schools.com/css/tryit.asp?filename=trycss3_resize_width
- ✓ La propiedad **outline-offset** agrega espacio entre un contorno y el borde o borde de un elemento.
 - https://www.w3schools.com/css/tryit.asp?filename=trycss3_outline-offset

Este div tiene un contorno 15px fuera del borde del borde.

4. Propiedades de CSS avanzadas

4.12. Propiedad *Box-sizing*:

- ✓ Por defecto, el ancho y el alto de un elemento se calcula así:
 - $\text{ancho} + \text{relleno} + \text{borde} = \text{ancho real de un elemento}$
 - $\text{alto} + \text{relleno} + \text{borde} = \text{altura real de un elemento}$
 - Cuando se establece el ancho / alto de un elemento, el elemento a menudo parece más grande de lo que se ha establecido (porque el borde y el relleno del elemento se agregan al ancho / alto especificado del elemento).
- ✓ **Box-sizing:** permite incluir el relleno y el borde en el ancho y alto total de un elemento.
- ✓ https://www.w3schools.com/css/tryit.asp?filename=trycss3_box-sizing_new

4. Propiedades de CSS avanzadas

4.13. Media Queries:

- ✓ Una de las herramientas esenciales para poder realizar un diseño responsivo de calidad son las **media-queries** que son: “... módulo de CSS3 que nos permite adaptar al representación del contenido a las características del dispositivo...” (Definición de Wikipedia)
- ✓ Las media queries son expresiones en las que indicamos un tipo de medio y una consulta en relación a las características del dispositivo como alto, ancho e incluso color:

```
@media not|only mediatype and (expressions) {  
    CSS-Code;  
}
```

- ✓ https://www.w3schools.com/cssref/css3_pr_mediaquery.asp

4. Propiedades de CSS avanzadas

4.13. Media Queries:

- ✓ Tipos de valores para *mediatype*:

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

- ✓ Ejemplo: https://www.w3schools.com/css/tryit.asp?filename=trycss3_media_queries1

4. Propiedades de CSS avanzadas

4.13. Media Queries:

- ✓ Condiciones que se pueden consultar:
 - width | min-width | max-width
 - height | min-height | max-height
 - orientation (landscape / portrait)
 - aspect-ratio | min-aspect-ratio | max-aspect-ratio
 - color | min-color | max-color

- ✓ Ejemplo: https://www.w3schools.com/css/tryit.asp?filename=trycss3_media_queries1

4. Propiedades de CSS avanzadas

4.13. Media Queries:

✓ Ejemplo:

```
/* Estilos para todo tipo de pantallas con una anchura máxima de 576px*/
@media all and (max-width: 576px) {
    .....;
}

/* Estilos para pantallas con al menos 992px de anchura y que estén apaisadas (más ancho que alto)*/
@media screen and (min-width: 992px) and (orientation: landscape) {
    .....;
}

/* Estilos sólo para pantallas que tengan al menos 768px de anchura*/
@media only screen and (min-width: 768px) {
    .....;
}
```