

***RECUERDA PONER A GRABAR LA  
CLASE***





Clase 08. DESARROLLO WEB

# ***ANIMACIONES, TRANSFORMACIONES Y TRANSICIONES***



## ***OBJETIVOS DE LA CLASE***

- Crear animaciones con CSS.
- Agregar transformaciones a elementos.
- Incorporar transiciones a elementos.

# GLOSARIO:

## Clase 7

**CSS Grid:** es el sistema de maquetación más potente que hay disponible. Se trata de un sistema en 2D que permite definir filas y columnas (a diferencia de, por ejemplo, Flexbox, el cual funciona en una única dimensión).

**Diseño responsive:** se refiere a la idea de que un sitio web debería mostrarse igual de bien en todo tipo de dispositivo, desde monitores de pantalla panorámica hasta teléfonos móviles. El diseño responsive se logra a través de "Media Queries" de CSS. Pensemos en las Media Queries como una forma de aplicar condicionales a las reglas de CSS.

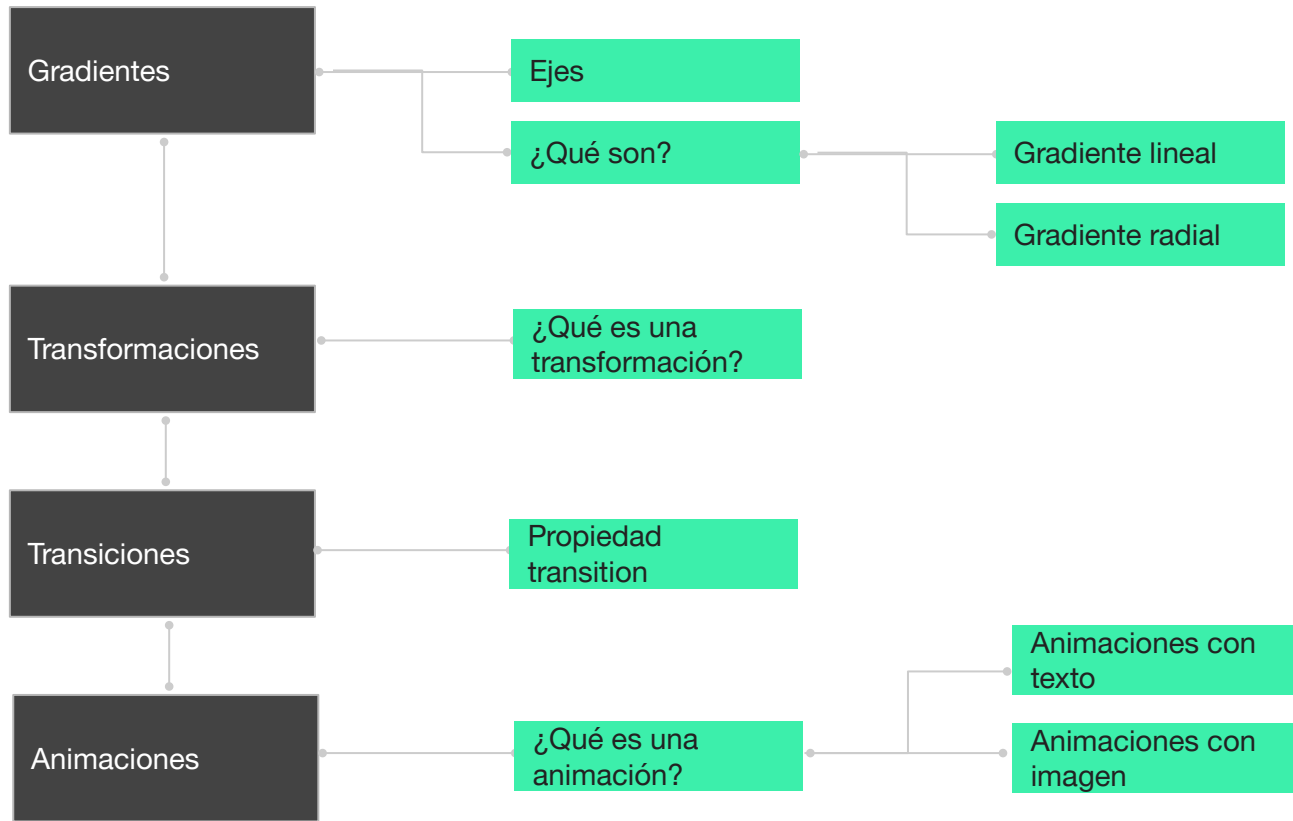
**Mobile First:** significa crear el código primero para los dispositivos más pequeños que los usuarios probablemente tengan, como teléfonos o tabletas. Implica trabajar en el dispositivo más pequeño, y luego acumular desde allí todo en el mismo código y el mismo proyecto, en lugar de hacer uno nuevo para cada tamaño de pantalla.

**Meta viewport:** una etiqueta <meta> viewport da al navegador las instrucciones sobre cómo controlar las dimensiones, y el ajuste a escala de la página.

# ***MAPA DE CONCEPTOS***

# MAPA DE CONCEPTOS CLASE 8

¡Para  
recordar!



# ***CRONOGRAMA DEL CURSO***

## Clase 7



### Grids v2



PRÁCTICAS DE LO  
VISTO EN CLASE

## Clase 8



### Animaciones, transformaciones y transiciones



PRÁCTICAS DE LO  
VISTO EN CLASE



APLICANDO GRIDS



FULL RESPONSIVE

## Clase 9



### GIT

TRANSFORMACIONES Y  
ANIMACIONES

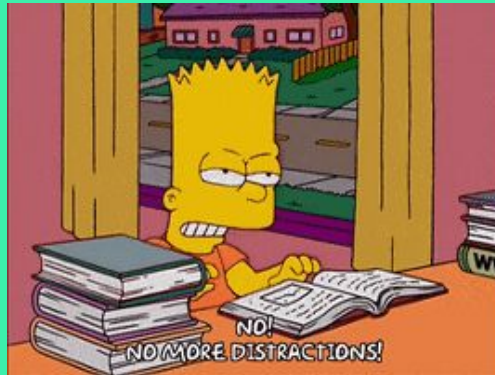


SEGUNDA ENTREGA  
DEL PROYECTO FINAL



# ***GUIÓN DE LA CLASE***

Accede al material complementario [aquí](#).



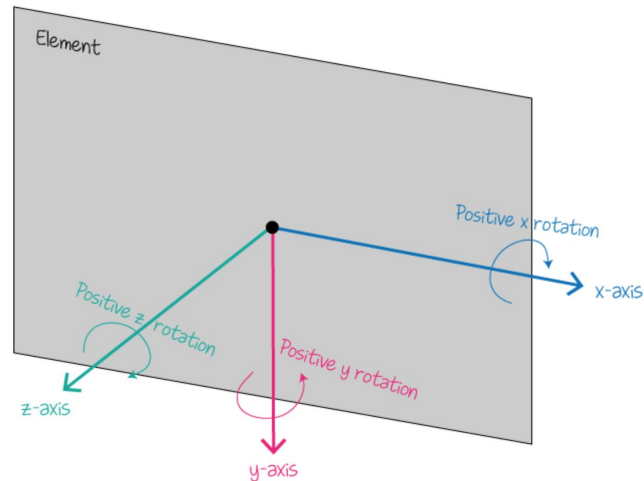


# ***GRADIENTES***

# EJES

Para entender los valores que debemos aplicar, es necesario comprender el concepto de los ejes.

- **X** se refiere a la posición horizontal, de izquierda a derecha.
- **Y** se refiere a la posición vertical, de arriba hacia abajo.
- **Z** puedes también mover los elementos **hacia adelante o atrás** en el documento (2D), como si se tratara de un espacio 3D.



# GRADIENTES

👉 **CSS3** ha agregado la opción de crear gradientes (fondos en degradé) sin la necesidad de usar imágenes.

👉 Los gradientes en CSS son de **dos tipos: lineales (*linear-gradient*) y radiales (*radial-gradient*)**.

👉 En el gradiente lineal, la transformación de color va avanzando línea a línea; mientras que en el radial, dicha transformación se produce debido a que sucesivos círculos concéntricos van cambiando de color.

# GRADIENTES



El gradiente normalmente se usa en la propiedad **background**:

👉 Gradientes lineales:

```
.clase {  
  background-image: linear-gradient(to left, red , yellow);  
}
```

Puedes elegir el punto de inicio del gradiente. Los puntos de inicio pueden ser **top, right, left o bottom** de tu caja, o puedes escoger los grados de inclinación que quieres que tenga tu gradiente.



# GRADIENTES

Valor por defecto

## HTML

```
<div id="grid">  
  Lorem ipsum dolor  
  sit amet,  
  consectetur  
  adipiscing elit.  
</div>
```

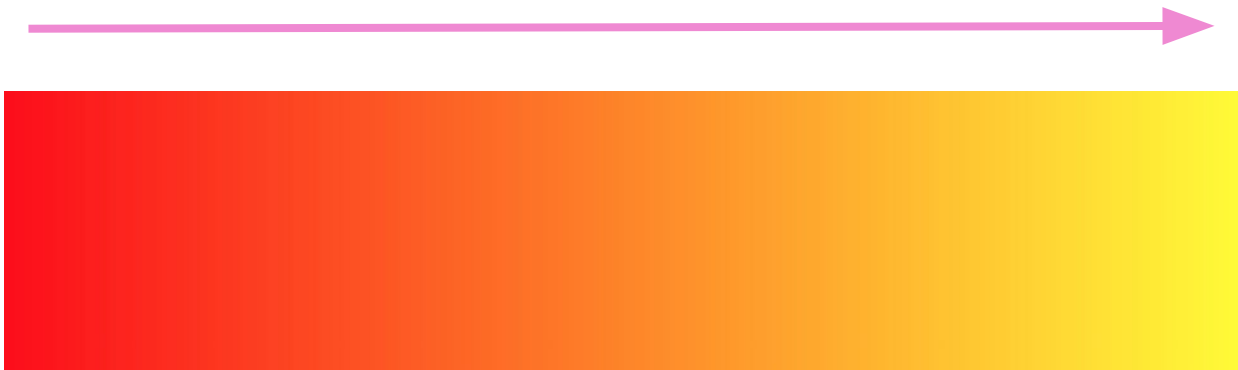


Lorem ipsum dolor sit amet, consectetur adipiscing elit.

```
#grad {  
  background-image: linear-gradient(red, yellow);  
}
```

Forma predeterminada: de arriba hacia abajo.

# GRADIENTES



```
#grad {  
  background-image: linear-gradient(to right, red , yellow);  
}
```

Hacia la derecha, desde el rojo hacia el amarillo.

# ***GRADIENTES***



```
#grad {  
  background-image: linear-gradient(to bottom, red , yellow);  
}
```

Hacia abajo, desde el rojo hacia el amarillo.

# ***GRADIENTES***



```
#grad {  
  background-image: linear-gradient(to left, red , yellow);  
}
```

Hacia la izquierda, desde el rojo hacia el amarillo.



# ***GRADIENTES***



```
#grad {  
  background-image: linear-gradient(to top, red , yellow);  
}
```

Hacia arriba, desde el rojo hacia el amarillo.

Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***

# ***TRANSFORMACIONES***

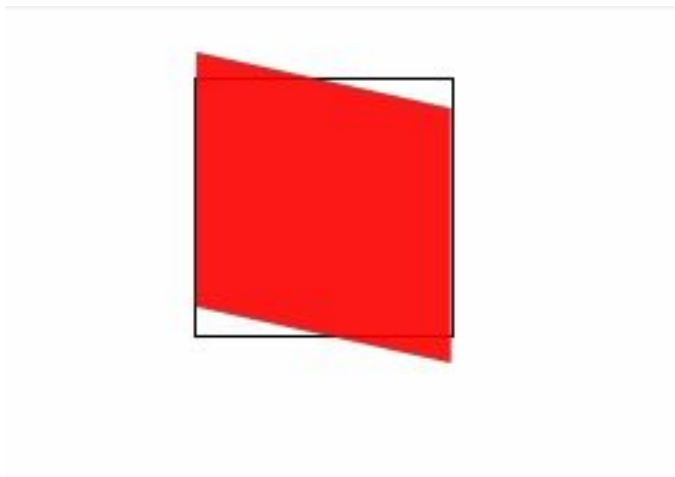
# ***TRANSFORMACIONES***

Una transformación es una modificación de la forma en que se muestra un elemento. Todo elemento transformado por CSS cambia la forma en que se ve, pero no el lugar que ocupa. Los efectos que se pueden lograr son:

- Mover un elemento de lugar (¡sin position!).
- Escalar el tamaño de un elemento.
- Voltear y girar elementos.
- Cambiar la perspectiva de un elemento.

# TRANSFORMACIONES

Los siguientes **ejemplos** que veremos están animados (tema que veremos más adelante), para un entendimiento más simple.



# ***TRASLADAR-MOVER OBJETOS***

*transform:translate( )* cambia la ubicación del objeto (como si fuese un position).

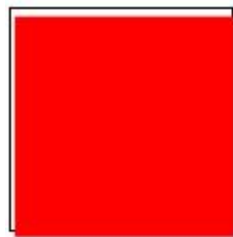
Requiere dos números y su unidad, separados por una coma:

- El primero es el desplazamiento horizontal (eje X).
- El segundo el desplazamiento vertical (eje Y).
- Valores positivos mueven a la derecha/abajo.
- Valores negativos mueven a la izquierda/arriba.
- Sí, existe `translateX()` y `translateY()`, cada uno sólo recibe un número con su unidad.

# ***TRANSFORM: TRANSLATE***

Valor: **translate**, especificamos los eje X e Y a donde queremos que se mueva el elemento.

```
div {  
    transform: translate(10px, 20px);  
}
```



# ***ROTACIÓN DE OBJETOS***

La rotación permite girar un objeto sin deformarlo. Se hace con el `transform: rotate()`. Recibe entre paréntesis un número que representa la cantidad de grados a girar el objeto:

- Si es **positivo**, rota hacia la **derecha** (en sentido horario).
- Si es **negativo**, rota hacia la **izquierda** (sentido antihorario).

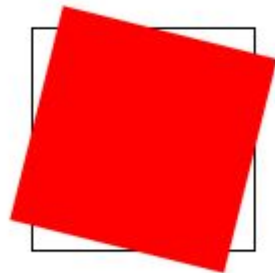
Por tratarse de grados, la unidad que acompaña el número será `deg` (degrees).



# ***TRANSFORM: ROTATE***

Valor: rotate, se especifican los grados a rotar (máximo 360).

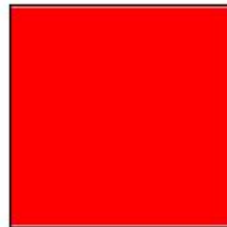
```
div {  
    transform: rotate(360deg);  
}
```



# ***TRANSFORM: ROTATEX***

Valor: rotateX, rotar en X, especificando los grados.

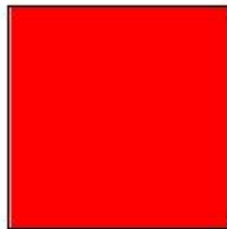
```
div {  
  transform: rotateX(70deg);  
}
```



# ***TRANSFORM: ROTATEY***

Valor: rotateY, rotar en Y, especificando los grados.

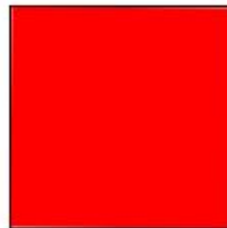
```
div {  
    transform: rotateY(70deg);  
}
```



# ***TRANSFORM: ROTATEZ***

Valor: **rotateZ**, rotar en Z, especificando los grados.

```
div {  
  transform: rotateZ(90deg);  
}
```



# ***ESCALAR OBJETOS***

`transform:scale( )`, cambia la escala del objeto (como si fuese un zoom).

Requiere dos números separados por coma:

- El primero es el ancho (Escala en eje X).
- El segundo el es alto (Escala en eje Y).
- Valores mayores a 1, agrandan.
- Valores entre 1 y 0, achican.
- Valores negativos, escalan dado vuelta.
- Si solo se quiere cambiar un eje, existe `scaleX()` y `scaleY()`, cada uno solo recibe un número.

# ***TRANSFORM: SCALE***

Valor: **scale**, cuantas veces se va a escalar (*alto, ancho*), se puede poner un único valor.

```
div {  
    transform: scale(2,1);  
}
```



# ***SESGAR ELEMENTO***

`transform:skew()`, para deformar objetos en el CSS utilizamos el método `skew` (sesgar).

Puede tener hasta dos números separados por coma:

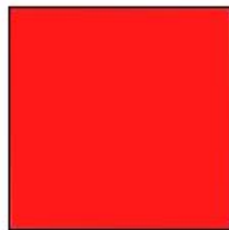
- Sus parámetros son los ángulos de deformación en grados sexagesimales (**deg**).
- El primero indica el eje **“X”**.
- El segundo indica el eje **“Y”**.

\*Sesgar: tocer.

# ***TRANSFORM: SKEW***

Valor: **skewX**, perspectiva en ambos ejes (X,Y).

```
div {  
    transform: skew(20deg,10deg);  
}
```

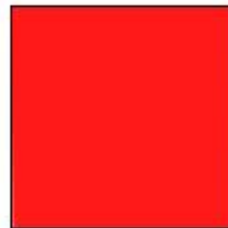




# ***TRANSFORM: SKEWX***

Valor: **skewX**, perspectiva en X.

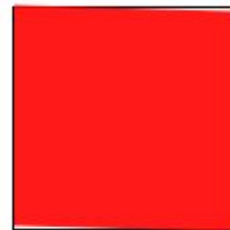
```
div {  
    transform: skewX(20deg);  
}
```



# ***TRANSFORM:- SKEWY***

Valor: **skewY**, perspectiva en Y.

```
div {  
    transform: skewY(20deg);  
}
```



# ¿RECUERDAN EL EJEMPLO?

Así empezamos...



Importante!

## HTML

```
<!-- Caja Roja -->  
<div>  
</div>
```

## CSS

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: rgba(255,0,0,0.9);  
  transform: skewY(25deg)  
}
```



Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***

# ***TRANSICIONES***

# TRANSICIONES

Con la propiedad **transition**, es posible lograr que al pasar el mouse por el elemento, el mismo “haga una animación”.

Recuerda que para los enlaces se utiliza `a:hover`, con el fin de que cambien sus estilos al pasar el mouse por encima. `:hover` se puede utilizar con **cualquier** elemento sobre el cual quisieras ejecutar una transición, un div, span, párrafo, etc.

**Veamos los ejemplos...**

# TRANSICIONES

Imaginemos que queremos que cambie su altura: debemos indicar **qué propiedad** queremos que se anime y **por cuántos segundos** (2 segundos).

```
div {  
    height: 100px;  
    transition: height 2s;  
    /* propiedad duración */  
}  
  
div:hover {  
    height: 200px;  
}
```




Contenido

# TRANSICIONES

También se puede especificar más de una propiedad:

```
div {  
  width: 100px;  
  height: 100px;  
  transition: height 2s, width 1s;  
}  
div:hover {  
  height: 200px; width: 200px;  
}
```



Contenido



# TRANSICIONES

Aplicar transición **a toda propiedad** que haya variado:

```
div {  
    width: 100px; height: 100px;  
    transition: all 2s;  
}  
div:hover {  
    height: 200px; width: 200px;  
    padding: 20px;  
    background-color: cyan;  
}
```

Contenido

## ¿Sólo con la propiedad “:hover” funciona?

**No**, también lo hace con cualquier propiedad del elemento que aplique cambios en él.

Por ejemplo con la propiedad **focus**, la cual indica que el elemento tiene el foco. Generalmente se activa cuando el usuario hace clic, toca un elemento o lo selecciona con la tecla "Tab" del teclado.

```
input {  
  width: 100px; height: 100px;  
  transition: all 2s;  
}  
  
input:focus {  
  width: 200px;  
  font-size: 24px;  
}
```



# ¿RECUERDAN EL EJEMPLO?

Ahora lo transformamos

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: rgba(255,0,0,0.9);  
  border: solid 1px;  
}  
div:hover {  
  padding: 20px;  
  transform: skewY(25deg)  
}
```



Importante!



**CODER HOUSE**

# ***ANIMACIONES***



# ANIMACIONES

- 👉 A diferencia de la transición, una animación es un efecto que se loopea tantas veces como se quiera.
- 👉 No depende del cambio de estado (el elemento se animará desde la carga de la web).
- 👉 **Es la unión de dos partes:** por un lado, una línea de tiempo (llamada keyframe) con la información de los cambios; por otro, aplicar ese keyframe a un elemento que será el que se verá animado.

# LINEA DE TIEMPO



- Es un elemento `@keyframes` *leo* {*aca iria el codigo css*} con un nombre. Luego del nombre y entre llaves, se definen los puntos donde cambiará el CSS.
- Cada cambio pasa en un porcentaje de la animación. Por cada punto de inflexión, y entre llaves, van las reglas CSS que se aplicarán en ese momento.
- El cambio es paulatino de un porcentaje al otro.

# LINEA DE TIEMPO



```
@keyframes un_efecto {  
  0%{ width: 100px; }  
  0.2s{ width: 50px; }  
  25%{ width: 300px; }  
  50%{ width: 200px;  
    background-color: red; }  
  75%{ width: 300px;}  
  100%{ width: 100px;  
    background-color:green;}  
}
```

```
section {  
  width: 100px;  
  height: 100px;  
  background-color: green;  
  animation-name: un_efecto;  
  animation-iteration-count: infinite;  
  animation-timing-function: ease-in;  
  animation-duration: 2s;  
  animation-delay: 10s;  
}
```



# ***EJEMPLO***



ejemplo

# ¿RECUERDAN EL EJEMPLO?

Ahora lo animamos...

Importante!



```
div.animame {  
  width: 100px; height: 100px;  
  background-color: rgba(255,0,0,0.9);  
  border: solid 1px;  
}  
div {  
  position: relative;  
  Left:200px; top:50px;  
}
```

```
div:not(.animame) {  
  border: solid 1px;  
  width: 100px;  
  height: 100px;  
  position: absolute;  
  left:207px;  
  top:57px;  
}
```

```
@keyframes animacion {  
  from { }  
  to {  
    /* probar propiedades */  
    transform: skewY(25deg)  
  }  
} /* configuracion inicial */  
.animame {  
  animation: animacion 1s linear 3s infinite alternate;}
```

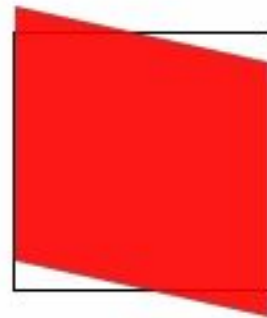
# ¿RECUERDAN EL EJEMPLO?

Ahora lo animamos...



HTML

```
<!-- Caja Blanca -->  
<div>  
</div>  
<!-- Caja Roja -->  
<div class="animame">  
</div>
```



Ejemplo  
en vivo



***¡VAMOS A PRACTICAR LO VISTO!***



Puedes hacer animaciones con texto:

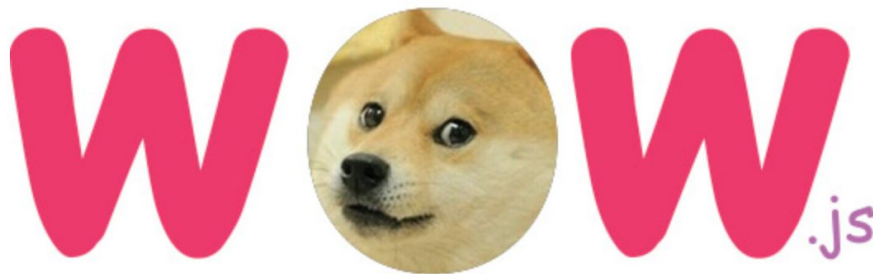
```
h1 {  
  animation-duration: 2s;  
  animation-name: aparecer;  
  animation-iteration-count: infinite;  
}  
  
@keyframes aparecer {  
  0% {  
    opacity: 0;  
  }  
  100% {  
    opacity: 1;  
  }  
}
```

**Titulo**

# LINEA DE TIEMPO



Puedes usar lo siguiente para poner las animaciones que quieras,  
haz clic sobre la imagen para ver la página web:



Reveal Animations When You Scroll. Very [Animate.css](#) Friend :-)  
Easily customize animation settings: style, delay, length, offset, iterations...  
100% MIT Licensed, not GPL. Keep your code yours.  
ES2015+, naturally Caffeine free.

# ¿CÓMO LO INSTALO?



1

Enlace a la biblioteca de animación CSS (dentro de **<head>**).

```
<link rel="stylesheet"
href="https://raw.githubusercontent.com/daneden/animate.css/master/animate.css">
```

2

Enlace y activar wow.js (poner antes del cierre de **<body>**).

```
<script src="js/wow.min.js"></script>
<script>
    new WOW().init();
</script>
```

# ¿CÓMO LO USO?



1

## Haz un elemento revelable

Agregue la clase CSS `.wow` a un elemento HTML: será invisible hasta que el usuario se desplace para revelarlo.

```
<div class="wow tipoAnimacion">  
    Contenido para revelar aquí  
</div>
```

2

## Elige el estilo de animación

Elija un estilo de animación en `Animate.css`, luego agregue la clase CSS al elemento HTML.

```
<div class="wow bounceInUp">  
    Contenido para revelar aquí  
</div>
```





# ***APLICANDO GRIDS***

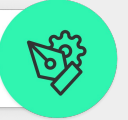
Aplica GRIDS + Flexbox en una página de tu sitio web

# APLICANDO GRIDS + Flexbox

**Formato:** Archivo html y css

**Sugerencia:** Carpeta en formato zip o rar con el/los archivos html y css.

Desafío  
entregable



## >> Consigna:

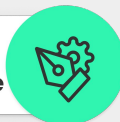
Generar que el index y una página más a elección de nuestro proyecto sea totalmente responsive utilizando grids para el layout, flexbox para los componentes y box modeling para terminar de acomodar los elementos. Es necesaria la utilización de media queries.

# APLICANDO GRIDS + Flexbox

**Formato:** Archivo html y css

**Sugerencia:** Carpeta en formato zip o rar con el/los archivos html y css.

Desafío  
entregable



## >> Recomendaciones:

- En el html, generar estructura de grid-contenedor-padre e grid-item-hijo para poder trabajar desde el CSS con grid-area.
- Dentro de esos grid-item-hijo deberemos agregar etiquetas para generar los componentes (ej: nav - footer - content - etc) a los cuales acomodaremos aplicando flexbox.
- Si es necesario, aplicamos box modeling para terminar de acomodar y generar nuestro layout completo para la vista desktop.
- Deberás repetir este proceso pero dentro de una media query mobile.



# ***FULL RESPONSIVE***

Generar que todo nuestro proyecto sea totalmente responsive para desktop y mobile.

# TRANSFORMACIONES Y ANIMACIONES

**Formato:** archivo HTML y CSS. Debe tener el nombre "Idea+Apellido".

**Sugerencia:** carpeta en formato zip o rar con el/los archivos html y css.

Desafío  
Complementario



## >> Consigna:

Generar todo nuestro **proyecto responsive**, utilizando grids para el layout, flexbox para los componentes y box modeling para terminar de acomodar los elementos.

Es necesaria la utilización de media queries.

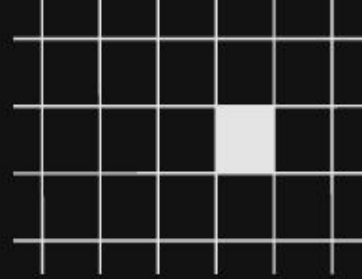
***¿PREGUNTAS?***



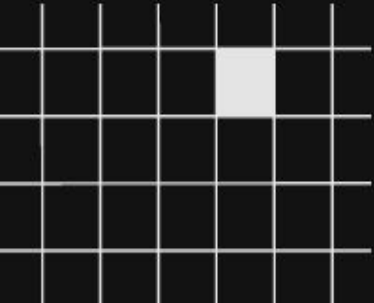
***¡DESCUENTO EXCLUSIVO!***



[Quiero saber más](#)



*¡Completa tu carrera y potencia tu desarrollo profesional!*  
*Ingresando el cupón **CONTINUATUCARRERA** tendrás un*  
*descuento para inscribirte en el próximo nivel. Puedes*  
*acceder directamente desde la plataforma, entrando en la*  
*sección ["Cursos y Carreras"](#).*

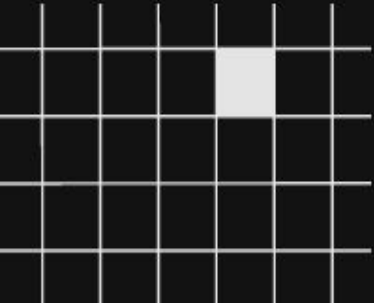






# ***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- Crear animaciones con CSS.
  - Agregar transformaciones a objetos.
  - Agregar transiciones a elementos.
- 

***#DEMOCRATIZANDO LA EDUCACIÓN***

***CODER HOUSE***