



Formularios

Cambiar el ancho de los campos

- Mediante estilos CSS se puede mejorar considerablemente el estilo de los formularios.
- Podemos determinar el ancho de los campos de tipo input con la propiedad width

```
input {  
    width: 100%;  
}
```

- El ejemplo anterior se aplicará a todos los elementos de tipo `<input>`. Si queremos aplicar el estilo sólo a un tipo específico de elemento input, deberemos indicarlo mediante el selector adecuado.
 - `input[type=text]` – Sólo afectará a los campos de tipo texto
 - `input[type=password]` – Sólo afectará a los campos de tipo password
 - `input[type=number]` – Sólo afectará a los campos de tipo number
- ```
input[type=text], input[type=email], input[type=url] {
 width: 100%;
}
```

# Padding

- Podemos usar la propiedad de padding para añadir espaciado dentro de los campos

```
input {
 padding: 12px;
}
```

- Al igual que en el caso anterior, se puede añadir padding a todos los campos de tipo input o a los campos de un tipo específico.

```
input[type=text], input[type=email], input[type=url] {
 padding: 10px 5px;
}
```

- También podemos añadir márgenes para añadir espacio alrededor de la caja del campo

```
input[type=text], input[type=email], input[type=url] {
 margin: 5px 10px 10px 10px;
}
```

# Bordes

- Podemos usar la propiedad `border` para definir los atributos de los bordes de los campos del formulario

```
input[type=text] {
 border: 2px solid red;
 border-radius: 4px;
}
```

- La propiedad `border-radius` hace que las esquinas de la caja tomen forma redondeada
- Si por ejemplo, sólo queremos un borde podemos aplicar la propiedad correspondiente:

```
input[type=text] {
 border: none;
 border_bottom: 2px solid red;
}
```

# Color de fondo

- Con la propiedad background-color podemos definir un color de fondo para los campos

```
input[type=text] {
 background-color: #3CBC8D;
 color: white;
}
```

# Formetear campo activo

○ Por defecto, algunos navegadores añaden una línea exterior en el campo cuando esté esta "enfocado", es decir, cuando hemos hecho clic sobre él.

○ Podemos quitar esta línea con la propiedad `outline:none`

```
input[type=text] {
 outline:none;
}
```

○ También se puede cambiar el formato del campo activo mediante el selector `:focus`

```
input[type=text]:focus {
 border: 3px solid #555;
 background-color: lightblue;
}
```

# Campo con icono interior

- En ocasiones deseamos mostrar una pequeña imagen en el interior de uno de los campos de formulario.
- Para ello utilizaremos la propiedad background-image.

```
input[type=text] {
 background-image: url('searchicon.png');
 background-position: 10px 10px;
 background-repeat: no-repeat;
 padding-left: 40px;
}
```

# Formatos textarea

```
textarea {
 width: 100%;
 height: 150px;
 padding: 12px 20px;
 border: 2px solid #ccc;
 border-radius: 4px;
 background-color: #f8f8f8;
 resize: none;
}
```



# Estilos select

```
select {
 width: 100%;
 padding: 16px 20px;
 border: none;
 border-radius: 4px;
 background-color: #f1f1f1;
}
```

# Estilos botones input

```
0 input[type=button], input[type=submit],
 input[type=reset] {
 background-color: #4CAF50;
 border: none;
 color: white;
 padding: 16px 32px;
 text-decoration: none;
 margin: 4px 2px;
}
```



Tablas

# Introducción

- Ya hemos visto que por defecto una tabla html no muestra los bordes que delimitan cada celda.

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

- También vimos que podíamos hacer que este borde se volviera visible añadiendo el atributo border a la etiqueta <table>

<table border="1">

En este caso, se mostrará un borde doble alrededor de cada celda:

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

- Podemos cambiar éstos y otros aspectos de las tablas utilizando los estilos css, a través de los cuales podremos conseguir tablas mucho más vistosas

# Bordes de tabla

○ Para especificar los bordes de una tabla en CSS, utilizaremos la propiedad `border`

○ Sintaxis:

```
border: grosor estilo color;
```

Donde:

○ `grosor` es el grosor del borde. Podemos especificarlo, por ejemplo, en píxeles.

○ `estilo`. Es el estilo de borde (podemos usar los valores que ya hemos visto para otros bordes)

○ `color` es el color de borde especificado en cualquiera de los modos vistos.

○ Podemos definir un estilo de borde para la tabla o para las celdas.

## Código

```
table {
 border: dashed red;
}
```

```
td {
 border: dashed red;
}
```

```
table, td {
 border: dashed red;
}
```

Si tenemos filas de título con <th> y definimos los estilos como mostramos arriba los bordes se verían como se muestra a la derecha

```
table, td, th {
 border: dashed red;
}
```

## Visualización

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

Observamos que la tabla superior tiene bordes dobles. Esto es debido a que tanto la tabla como los elementos <th> y <td> tienen fronteras separadas.

# border-collapse

- Esta propiedad permite determinar si los bordes se deben fusionar en un borde simple y cuando se deben mantener como bordes separados.
- Su sintaxis es la siguiente:

`border-collapse: valor;`

Donde valor puede ser:

- `collapse`. fusiona de forma automática los bordes de las celdas adyacentes
- `separate`. Fuerza a que cada celda muestre sus cuatro bordes lo que provoca ese efecto de doble borde cuando se pone el borde a dos celdas contiguas.
- El valor por defecto es `separate`, aunque generalmente, `collapse` suele ser mucho más claro



```
table {
 border-collapse: collapse;
}
table, td, th {
 border: 1px solid black;
}
```

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

Si sólo queremos poner un borde a la tabla, no necesitaremos la propiedad `border-collapse`, con la propiedad `border` será suficiente.

# border-spacing

- La propiedad border-spacing define la distancia entre los bordes de las celdas adyacentes (sólo para el modelo de "fronteras separadas").

```
table {
 border-collapse: separate;
 border-spacing: 10px 50px;
}
```

- Su sintaxis es la siguiente:

**border-spacing:** *length* [*length*]|initial|inherit;

- *Length* [*length*]. Especifica la distancia entre los bordes de las celdas adyacentes en píxeles, cm, etc. Los valores negativos no están permitidos
  - Si se especifica un solo valor, éste especificará tanto el espacio horizontal como el vertical
  - Si se especifican dos valores de longitud, el primero establece el espaciado horizontal y el segundo establece el espaciado vertical



# Color de fondo y de texto

- o La propiedad background-color y color nos permiten establecer el color de fondo y del texto de los diferentes elementos de la tabla.
- o Estas propiedades las podemos utilizar con las etiquetas <table>, <tr>, <th> o <td>



```
th {
 background-color: #4CAF50;
 color: white;
}
```

# Ancho y alto de la tabla

- 0 El ancho y alto de la tabla se define con las propiedades width y height.

```
table {
 width: 100%;
}

th {
 height: 50px;
}
```

# Alineación horizontal

- Podemos establecer la alineación horizontal de los textos en las celdas mediante la propiedad `text-align`
- Su sintaxis es la siguiente:  
`text-align: alineación;`
  - Alineación: puede tomar los valores `left`, `right` o `center`
- Podemos definir alineación horizontal para la etiqueta `table`, `tr`, `td` o `th`.
- Por defecto, los elementos `th` ya aparecen centrados y los elementos `td` alineados a la izquierda.

# Alineación vertical

- o La propiedad vertical-align nos permite definir la alineación vertical del texto que se encuentra dentro de las celdas.
- o Su sintaxis es la siguiente:

`vertical-align: alineación;`

Donde alineación puede tomar los valores `top`, `middle` o `bottom`

- o La alineación vertical por defecto es middle.



```
td {
 height: 50px;
 vertical-align: bottom;
}
```

El ejemplo anterior, establece una altura de 50px para las celdas y una alineación inferior.

# Padding

- El espacio entre los bordes de las celdas y el contenido de las mismas se establece con la propiedad padding.
- Esta propiedad debe establecerse en las etiquetas <td> o <th>



```
th, td {
 padding:5px;
}
```

# Divisores horizontales

- En ocasiones no queremos poner todos los bordes de la tabla, sino que simplemente queremos que se vea un divisor horizontal que delimite lo que ocupa cada fila.
- Esto se puede hacer añadiendo la propiedad `border-bottom` a las etiquetas `<th>` o `<td>`



```
th, td {
 border-bottom: 1px solid #ddd;}
```

| Nombre | Apellidos | DNI          |
|--------|-----------|--------------|
| Luis   | García    | 11.011.456-J |
| Pedro  | Pérez     | 10.415.258-X |

# Otros formatos

- Podemos utilizar el selector `:hover` en las etiquetas `<tr>` para resaltar filas de la tabla al pasar el mouse por encima.

`tr:hover {background-color: #f5f5f5}`

- Podemos generar una tabla con filas rayadas utilizando el selector `nth-child` y añadiendo un color de fondo para todas las filas pares (o impares) de la tabla.

`tr:nth-child(even) {background-color: #f2f2f2}`

- Pondremos `even` si queremos aplicar el color a las filas pares y pondremos `odd` si queremos aplicar el color a las filas impares
- El título de las tablas se establece mediante el elemento `<caption>`, que por defecto se muestra encima de los contenidos de la tabla. La propiedad `caption-side` permite controlar la posición del título de la tabla.

`caption-side: top|bottom|initial|inherit;`

- `Empty-cells`. Esta propiedad nos permite definir el tratamiento que tendrán las celdas vacías de la tabla.

`empty-cells: show|hide|initial|inherit;`

- El valor `hide` indica que las celdas vacías no se deben mostrar. El valor `show` indica que debe mostrarse el fondo y los bordes de las celdas vacías (este es el valor por defecto)



Bordes  
redondeados



# Bordes redondeados

o Con la propiedad `border-radius` podemos redondear las esquinas de cualquier elemento.

o Sintaxis:

`border-radius: 1-4 Length|% |initial|inherit;`

o 1-4 son de 1 a 4 valores que representan los cuatro bordes (superior-izquierdo, superior-derecho, inferior-derecho e inferior-izquierdo).

o `length`. Define la unidad de medida en la que se están mostrando los valores

o `%`. Define la forma de la esquina en %

```
div {
 border: 2px solid;
 border-radius: 25px;
}
```

o La propiedad `border-radius` es la shorthand de las propiedades `border-top-left-radius`, `border-top-right-radius`, `boder-bottom-right-radius` y `border-bottom-left-radius`

# Bordes redondeados para cada esquina

- Se puede establecer un redondeado de esquina diferente para cada borde de un elemento.

```
#rcorners4 {
 border-radius: 15px 50px 30px 5px;
 background: #73AD21;
 padding: 20px;
 width: 200px;
 height: 150px;
}
```



- `border-radius: 15px 50px 30px`



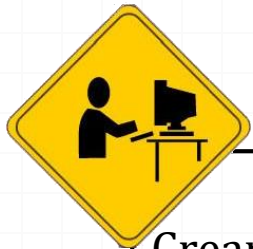
- `border-radius: 15px 50px`



- `border-radius: 50%`



- o La propiedad border-radius es una propiedad shorthand para definir una curvatura en las esquinas de la caja, pero existen las propiedades unitarias para cada borde:
  - o border-top-left-radius
  - o border-top-right-radius
  - o border-bottom-right-radius
  - o border-bottom-left-radius
- o Existe una sintaxis diferente para border-radius por navegador
  - o -moz-border-radius: 15px;



Crear una página web que contenga un div con un pequeño texto y una imagen.

Luego al div asignarle un color de fondo verde, un ancho de 200px y bordes redondeados de 25 píxeles

A la imagen asignarle un tamaño de 600 píxeles y un radio de 25px al borde inferior izquierdo



Colores

# Formatos de colores

- CSS permite definir los colores con sus nombres, formato hexadecimal o formato RGB.
- CSS3 introduce varias formas de definir los colores:
  - Colores RGBA
  - Colores HSL
  - Colores HSLA
  - Opacidad
- Colores RGBA
  - Son una extensión de los colores RGB donde a los tres valores del color RGB se añade un cuarto valor que llamamos canal alpha y que permite especificar la opacidad para el color.
  - EL formato de un color RGBA será:

`rgba(red, green, blue, alpha)`

    - alpha. Es un número entre 0.0 (completamente transparente) y 1.0 (totalmente opaco)

```
#p1 {background-color: rgba(255, 0, 0, 0.3);} /* red with opacity */
```

## ○ HSL

○ Hue, Saturation, Lightness (Tono, Saturación y Luminosidad).

○ Un color HSL se especifica con el formato:

`hsl(hue, saturation, lightness)`

○ Hue es un valor en el círculo cromático (desde 0 a 360, donde 0 es el rojo, 120 es verde y 240 es azul).

○ Saturation. es un porcentaje. 100% es el color completo.

○ Lightness es otro porcentaje (0% es oscuro (negro) y 100% es blanco).

```
#p1 {background-color: hsl(120, 100%, 50%);} /* verde*/
#p2 {background-color: hsl(120, 100%, 75%);} /* verde claro*/
#p3 {background-color: hsl(120, 100%, 25%);} /* verde oscuro*/
```

## ○ HSLA

○ Es una extensión de los colores HSL donde se ha añadido el canal alpha que permitirá especificar la opacidad para el color.

`hsl(hue, saturation, lightness, alpha)`

○ Alpha tiene el mismo significado y toma los mismos valores que en los colores RGBA

# Opacity

- Esta propiedad permite definir la opacidad para un color especificado en formato RGB.
- Su sintaxis es:

opacity: valor

- Valor. Tomará un valor entre 0.0 (totalmente transparente) y 1.0 (totalmente opaco)

```
/* Rojo con opacidad*/
#p1 {
 background-color:rgb(255,0,0);
 opacity:0.6;
}
```

- La diferencia entre utilizar opacity o utilizar los colores rgba está que con la primera forma el texto también se vuelve opaco.



Crear una página web que contenga dos párrafos, con id p1 y p2. Aplicar a p1 un color rgba(255,0,0,0.3) y al otro aplicar el color rgb(255,0,0) con opacidad de 0.3.  
Observar la diferencia



# Manos a la obra

Crear un documento HTML con cuatro párrafos de tipo H1.

- Al primero deberá aplicársele un color verde con una opacidad de 0.3 utilizando el formato rgba
- Al segundo deberá aplicársele un color verde utilizando la forma HSL (H=120, S=100%, L=50%)
- Al tercero deberá aplicársele el color verde del punto anterior pero con una opacidad de 0.3
- Al cuarto deberemos definir un color verde con una opacidad de 0.3 utilizando la propiedad `opacity`





Sombras

- o Con CSS3 se pueden añadir sombras a los textos y a los elementos.
- o Para añadir sombras contamos con dos propiedades:
  - o text-shadow
  - o box-shadow

# text-shadow

## o Sintaxis:

`text-shadow: h-shadow v-shadow blur-radius color|none|initial|inherit;`

- o `h-shadow`. Requerido. Define la posición horizontal de la sombra. Se pueden utilizar valores negativos.
- o `v-shadow`. Requerido. Define la posición vertical de la sombra. Se pueden utilizar valores negativos.
- o `blur-radius`. Opcional. El radio de desenfoque. El valor por defecto es 0
- o `Color`. Opcional. El color de la sombra. Se puede utilizar cualquiera de los formatos vistos, incluso definiendo opacidad.
- o `none`. Valor por defecto, significa sin sombra



Crear una página web que contenga tres párrafos, con id p1, p2 y p3. Aplicar a p1 una sombra con un desplazamiento de 2px y color rojo. A p2 aplicarle una sombra con un desplazamiento de -10px y color verde. En el tercer párrafo aplicarle una sombra con un desplazamiento de 2px y color azul pero sólo a la palabra sombra. Luego añadir un desenfoque de 5px a p2 y 10px a p3.

```
p1 {
 color: white;
 font-size: 2em;
 text-shadow: 2px 2px 3px #000000;
}
```

## ○ Efecto neón

```
p1 {
 Font-family: Arial;
 font-size: 2em;
 text-shadow: 2px 2px 3px #000000;
}
```

## ○ Sombras múltiples

```
p1 {
 Font-family: Arial;
 font-size: 2em;
 text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
}
```

## ○ Sombras múltiples

# box-shadow

o La propiedad box-shadow permite añadir una o más sombras a un elemento.

o Sintaxis:

```
box-shadow: inset h-shadow v-shadow blur spread color initial|inherit;
```

o spread. es el tamaño de la sombra. Es un valor opcional y se permiten valores negativos.

o color. Es el color de la sombra especificado en cualquiera de los formatos ya vistos. Por defecto se aplicará el color negro. En Safari este parámetro es obligatorio, si no se especifica el color, no se verá la sombra.

o inset. Este parámetro es opcional y si se utiliza permite cambiar de una sombra exterior (outset) a una sombra interior (inner).  
Este

## ◌ Elemento con sombra por defecto

```
div {
 box-shadow: 10px 10px;
}
```

## ◌ Elemento con sombra de un color

```
div {
 box-shadow: 10px 10px grey;
}
```

## ◌ Elemento con sombra de un color y desenfocada

```
div {
 box-shadow: 10px 10px 5px grey;
}
```

## ◌ Sombra interior

```
div {
 box-shadow: inset 5px 5px #888;
}
```



Bordes  
redondeados

# Bordes redondeados

o Con la propiedad `border-radius` podemos redondear las esquinas de cualquier elemento.

o Sintaxis:

`border-radius: 1-4 Length|% |initial|inherit;`

o 1-4 son de 1 a 4 valores que representan los cuatro bordes (superior-izquierdo, superior-derecho, inferior-derecho e inferior-izquierdo).

o `length`. Define la unidad de medida en la que se están mostrando los valores

o `%`. Define la forma de la esquina en %

```
div {
 border: 2px solid;
 border-radius: 25px;
}
```

o La propiedad `border-radius` es la shorthand de las propiedades `border-top-left-radius`, `border-top-right-radius`, `boder-bottom-right-radius` y `border-bottom-left-radius`



# Bordes redondeados para cada esquina

- Se puede establecer un redondeado de esquina diferente para cada borde de un elemento.

```
#rcorners4 {
 border-radius: 15px 50px 30px 5px;
 background: #73AD21;
 padding: 20px;
 width: 200px;
 height: 150px;
}
```



- `border-radius: 15px 50px 30px`



- `border-radius: 15px 50px`



- `border-radius: 50%`



- o La propiedad border-radius es una propiedad shorthand para definir una curvatura en las esquinas de la caja, pero existen las propiedades unitarias para cada borde:
  - o border-top-left-radius
  - o border-top-right-radius
  - o border-bottom-right-radius
  - o border-bottom-left-radius
- o Existe una sintaxis diferente para border-radius por navegador
  - o -moz-border-radius: 15px;



Crear una página web que contenga un div con un pequeño texto y una imagen.

Luego al div asignarle un color de fondo verde, un ancho de 200px y bordes redondeados de 25 píxeles

A la imagen asignarle un tamaño de 600 píxeles y un radio de 25px al borde inferior izquierdo



Colores

# Formatos de colores

- CSS permite definir los colores con sus nombres, formato hexadecimal o formato RGB.
- CSS3 introduce varias formas de definir los colores:
  - Colores RGBA
  - Colores HSL
  - Colores HSLA
  - Opacidad
- Colores RGBA
  - Son una extensión de los colores RGB donde a los tres valores del color RGB se añade un cuarto valor que llamamos canal alpha y que permite especificar la opacidad para el color.
  - EL formato de un color RGBA será:

`rgba(red, green, blue, alpha)`

    - alpha. Es un número entre 0.0 (completamente transparente) y 1.0 (totalmente opaco)

```
#p1 {background-color: rgba(255, 0, 0, 0.3);} /* red with opacity */
```

## ○ HSL

○ Hue, Saturation, Lightness (Tono, Saturación y Luminosidad).

○ Un color HSL se especifica con el formato:

`hsl(hue, saturation, lightness)`

○ Hue es un valor en el círculo cromático (desde 0 a 360, donde 0 es el rojo, 120 es verde y 240 es azul).

○ Saturation. es un porcentaje. 100% es el color completo.

○ Lightness es otro porcentaje (0% es oscuro (negro) y 100% es blanco).

```
#p1 {background-color: hsl(120, 100%, 50%);} /* verde*/
#p2 {background-color: hsl(120, 100%, 75%);} /* verde claro*/
#p3 {background-color: hsl(120, 100%, 25%);} /* verde oscuro*/
```

## ○ HSLA

○ Es una extensión de los colores HSL donde se ha añadido el canal alpha que permitirá especificar la opacidad para el color.

`hsl(hue, saturation, lightness, alpha)`

○ Alpha tiene el mismo significado y toma los mismos valores que en los colores RGBA

# Opacity

o Esta propiedad permite definir la opacidad para un color especificado en formato RGB.

o Su sintaxis es:

`opacity: valor`

o Valor. Tomará un valor entre 0.0 (totalmente transparente) y 1.0 (totalmente opaco)

```
/* Rojo con opacidad*/
#p1 {
 background-color:rgb(255,0,0);
 opacity:0.6;
}
```

o La diferencia entre utilizar opacity o utilizar los colores rgba está que con la primera forma el texto también se vuelve opaco.



Crear una página web que contenga dos párrafos, con id p1 y p2. Aplicar a p1 un color rgba(255,0,0,0.3) y al otro aplicar el color rgb(255,0,0) con opacidad de 0.3.

Observar la diferencia



# Manos a la obra

Crear un documento HTML con cuatro párrafos de tipo H1.

- Al primero deberá aplicársele un color verde con una opacidad de 0.3 utilizando el formato `rgba`
- Al segundo deberá aplicársele un color verde utilizando la forma HSL ( $H=120$ ,  $S=100\%$ ,  $L=50\%$ )
- Al tercero deberá aplicársele el color verde del punto anterior pero con una opacidad de 0.3
- Al cuarto deberemos definir un color verde con una opacidad de 0.3 utilizando la propiedad `opacity`





Sombras



- o Con CSS3 se pueden añadir sombras a los textos y a los elementos.
- o Para añadir sombras contamos con dos propiedades:
  - o text-shadow
  - o box-shadow

# text-shadow

## o Sintaxis:

`text-shadow: h-shadow v-shadow blur-radius color|none|initial|inherit;`

- o `h-shadow`. Requerido. Define la posición horizontal de la sombra. Se pueden utilizar valores negativos.
- o `v-shadow`. Requerido. Define la posición vertical de la sombra. Se pueden utilizar valores negativos.
- o `blur-radius`. Opcional. El radio de desenfoque. El valor por defecto es 0
- o `Color`. Opcional. El color de la sombra. Se puede utilizar cualquiera de los formatos vistos, incluso definiendo opacidad.
- o `none`. Valor por defecto, significa sin sombra



Crear una página web que contenga tres párrafos, con id p1, p2 y p3. Aplicar a p1 una sombra con un desplazamiento de 2px y color rojo. A p2 aplicarle una sombra con un desplazamiento de -10px y color verde. En el tercer párrafo aplicarle una sombra con un desplazamiento de 2px y color azul pero sólo a la palabra sombra. Luego añadir un desenfoque de 5px a p2 y 10px a p3.

```
p1 {
 color: white;
 font-size: 2em;
 text-shadow: 2px 2px 3px #000000;
}
```

## ○ Efecto neón

```
p1 {
 Font-family: Arial;
 font-size: 2em;
 text-shadow: 2px 2px 3px #000000;
}
```

## ○ Sombras múltiples

```
p1 {
 Font-family: Arial;
 font-size: 2em;
 text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
}
```

## ○ Sombras múltiples

# box-shadow

o La propiedad box-shadow permite añadir una o más sombras a un elemento.

o Sintaxis:

```
box-shadow: inset h-shadow v-shadow blur spread color initial|inherit;
```

o spread. es el tamaño de la sombra. Es un valor opcional y se permiten valores negativos.

o color. Es el color de la sombra especificado en cualquiera de los formatos ya vistos. Por defecto se aplicará el color negro. En Safari este parámetro es obligatorio, si no se especifica el color, no se verá la sombra.

o inset. Este parámetro es opcional y si se utiliza permite cambiar de una sombra exterior (outset) a una sombra interior (inner).  
Este

## ◌ Elemento con sombra por defecto

```
div {
 box-shadow: 10px 10px;
}
```

## ◌ Elemento con sombra de un color

```
div {
 box-shadow: 10px 10px grey;
}
```

## ◌ Elemento con sombra de un color y desenfocada

```
div {
 box-shadow: 10px 10px 5px grey;
}
```

## ◌ Sombra interior

```
div {
 box-shadow: inset 5px 5px #888;
}
```



Fonts

# @font-face

- En ocasiones utilizamos una fuente poco común y puede ocurrir que el usuario no la tenga instalada en su ordenador. En este caso, el navegador sustituirá por la letra de la familia genérica (si la hemos definido al especificar el font-family).
- La regla @font-face permite vincular fuentes sin necesidad de que el cliente las tenga instaladas en su ordenador.
  - Esta regla nos permite ubicar el fichero que contiene la fuente en nuestro servidor, luego asignaremos un nombre a esa fuente y cuando queramos hacer referencia a ella utilizaremos el nombre que le hayamos asignado.

```
@font-face {
 font-family: miFuente;
 src: url(wds052801.ttf);
}

div {
 font-family: miFuente;
}
```

- En el caso de utilizar font-face si queremos usar negritas o cursivas, es posible que tengamos que bajar e instalar en nuestro servidor el tipo de fuente con la especificación `_bold`, `_italic`, etc.
- A la hora de descargar fuentes, existen diferentes tipos de fuentes: TTF (True Type Fonts), OTF, WOFF, WOFF 2.0, SVG, EOT.
  - El formato más común son las TTF.
  - Algunos de los formatos anteriores no son soportados por todos los navegadores.



# @font-face embebida

- Se pueden utilizar fuentes alojadas en servidores externos al nuestro (como por ejemplo, google fonts).
- En google, teclear fonts google y escoger el enlace que pone google Fonts  
(o abrir directamente <https://www.google.com/fonts>)
- Buscar la fuente que se desea. A la izquierda existe un panel que nos permite especificar varios criterios para filtrar la búsqueda.
- Ir a quick-use.
- Dentro de quick-use nos aparecerá un enlace. Deberemos copiarlo en el head de la página.  

```
<link href='https://fonts.googleapis.com/css?family=Slabo+27px' rel='stylesheet' type='text/css'>
```
- Luego ya podemos utilizarlo dentro de font-family  

```
font-family:'slabo 27px';
```



# Transformaciones 2D

- o Las transformaciones CSS3 nos permiten trasladar, rotar, escalar y sesgar elementos.
- o Una transformación es un efecto que permite un cambio de elemento de forma, tamaño y posición.
- o CSS3 admite transformaciones 2D y 3D.
- o Para aplicar transformaciones utilizaremos la propiedad `transform`

# translate()

- El método `translate ()` mueve un elemento desde su posición actual (de acuerdo con los parámetros dados para el eje X y el eje Y).
- `translate(x,y)`. Define un movimiento 2D, moviendo el elemento a lo largo del eje X y del eje Y
- `translateX(n)`. Define un movimiento 2D, moviendo el elemento a lo largo del eje X
- `translateY(n)`. Define un movimiento 2D, moviendo el elemento a lo largo del eje Y
- En el siguiente ejemplo se mueve el elemento `<div>` 50 píxeles hacia la derecha, y 100 píxeles hacia abajo desde su posición actual:

```
div {
 -ms-transform: translate(50px,100px); /* IE 9 */
 -webkit-transform: translate(50px,100px); /* Safari */
 transform: translate(50px,100px);
}
```

# rotate()

- El método rotate () gira en sentido horario un elemento o hacia la izquierda de acuerdo con un determinado grado.
- rotate(angulo). Define una rotación 2D, donde angulo es el parámetro que especifica el ángulo de rotación.
- El ejemplo siguiente gira la <div> hacia la derecha con el elemento 20 grados:

```
div {
 -ms-transform: rotate(20deg); /* IE 9 */
 -webkit-transform: rotate(20deg); /* Safari */
 transform: rotate(20deg);
}
```

# scale()

- o El método `scale()` incrementa o decrementa el tamaño del elemento.
  - o `scale(x,y)`. Define una escala de transformación cambiando el ancho y el alto de los elementos.
  - o `scaleX(n)`. Define una escala de transformación, cambiando el ancho del elemento.
  - o `scaleY(n)`. Define una escala de transformación, cambiando el alto del elemento.

- o Ejemplo:

```
div {
 -ms-transform: scale(2,3); /* IE 9 */
 -webkit-transform: scale(2,3); /* Safari */
 transform: scale(2,3);
}
```

- o El ejemplo anterior, hace el div más grande. Si quisiéramos disminuir el tamaño en lugar de agrandarlo (por ejemplo a la mitad) usaríamos una escala de 0.5, 0.25, etc.

```
transform: scale(0.5,0.5);
```

# skew()

- o Este método permite sesgar un elemento.
  - o skewX(angulo). Permite sesgar un elemento a lo largo del eje X con el ángulo dado.
  - o skewY(angulo). Permite sesgar un elemento a lo largo del eje Y con el ángulo dado.
  - o skew(x-angulo,y-angulo). Permite sesgar un elemento a lo largo del eje X y del eje Y con los ángulos especificados.

## o Ejemplo 1

```
div {
 -ms-transform: skewX(20deg); /* IE 9 */
 -webkit-transform: skewX(20deg); /* Safari */
 transform: skewX(20deg);
}
```

## o Ejemplo 2

```
div {
 -ms-transform: skewY(20deg); /* IE 9 */
 -webkit-transform: skewY(20deg); /* Safari */
 transform: skewY(20deg);
}
```

## o Ejemplo 3

```
div {
 -ms-transform: skew(20deg, 10deg); /* IE 9 */
 -webkit-transform: skew(20deg, 10deg); /* Safari */
 transform: skew(20deg, 10deg);
}
```

# matrix()

- Este método permite combinar todos los métodos de transformación 2D en uno solo
- El método `matrix()` consta de seis parámetros, conteniendo los métodos que permiten rotar, escalar, mover y sesgar elementos.
- Sintaxis:

`matrix(n,n,n,n,n,n)`

`matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())`

- Ejemplo:

```
div {
 -ms-transform: matrix(1, -0.3, 0, 1, 0, 0); /* IE 9 */
 -webkit-transform: matrix(1, -0.3, 0, 1, 0, 0); /* Safari
 */
 transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```






# Manos a la obra

Crear un documento HTML con cuatro elementos de tipo <div> con las siguientes características:

- 100px de alto y 100px de ancho. Color azul claro. Borde sólido de 1px negro
- Mover el primer div 100px a la derecha y 200px hacia abajo.
- Rotar el segundo div 45 grados
- Cambiar el tamaño del tercer div a la mitad de su ancho, pero al doble de su alto
- Sesgar el cuarto div 20 grados sobre el eje X y 30 grados sobre el eje Y
- `-ms-transform: ; /* IE 9 */`  
`-webkit-transform: ; /* Safari */`  
`transform: ;`  
`translate(x,y) - rotate(xdeg) - scale(x,y) - skew(xdeg,ydeg)`



# Transformaciones 3D

- o CSS3 permite formatear elementos añadiendo transformaciones 3D
- o La propiedad transform admite los mismos valores vistos para el 3D solo que ahora podemos añadir el eje Z
- o rotate.rotateX(), rotateY(), rotateZ()
  - o El método rotateX() rota un elemento alrededor de sí mismo sobre el eje x con los grados indicados.

```
div {
 -webkit-transform: rotateX(150deg); /* Safari */
 transform: rotateX(150deg);
}
```
  - o El método rotateY() rota un elemento alrededor de sí mismo sobre el eje y con los grados indicados.

```
div {
 -webkit-transform: rotateY(130deg); /* Safari */
 transform: rotateY(130deg);
}
```
  - o El método rotateZ() rota un elemento alrededor de sí mismo sobre el eje z con los grados indicados.

```
div {
 -webkit-transform: rotateZ(130deg); /* Safari */
 transform: rotateZ(130deg);
}
```
- o translate3d(x,y,z) Define un movimiento 3D
- o scale3d(x,y,z) Define una modificación de la escala en 3D
- o rotate3d(x,y,z,angulo). Define una rotación 3d
- o perspective(n). Define una vista en perspectiva para una transformación 3D



# Manos a la obra

Crear un documento HTML con cuatro párrafos elementos de tipo `<div>` con las siguientes características:

- ◊ 100px de alto y 100px de ancho. Color azul claro. Borde sólido de 1px negro
- ◊ Rotar 150 grados alrededor del eje X.
- ◊ Rotar 120 grados alrededor del eje Y
- ◊ Rotar 90 grados alrededor del eje Z



Transiciones

- Las transiciones CSS3 en una propiedad nos permiten cambiar los valores iniciales a otros diferentes sin problemas durante un período determinado.
- Para crear un efecto de transición se deben especificar dos valores:
  - La propiedad CSS a la que se quiere añadir el efecto
  - La duración de este efecto. Si no se especificara este último valor, la transición no tendrá efecto porque por defecto este valor es 0.
- Para definir una transición:

1. Definir el formato original de la propiedad

```
div {
 width: 100px;
 height: 100px;
 background: red;
}
```

2. Dentro del div anterior, añadir la información referente a la propiedad que se va a ver afectada por la transición y al tiempo que deberá tomar la transición.

```
-webkit-transition: width 2s; /* Safari */
transition: width 2s;
```

3. Especificar el desencadenante de la transición así como el nuevo valor de la propiedad:

```
div:hover {
 width: 300px;
}
```

o Se pueden añadir transiciones a varias propiedades para que se lleven a cabo a la vez.

```
div {
 width: 100px;
 height: 100px;
 background: red;
 -webkit-transition: width 2s, height 4s; /*Safari 3.1 - 6.0 */
 transition: width 2s, height 4s;
}
```

```
div:hover {
 width: 300px;
 height: 300px;
}
```



# transition-timing-function

- Esta propiedad nos permite definir qué tipo de curva de velocidad usará la transición.
  - Por ejemplo, si tenemos una animación que mueve 10px un objeto durante 10 segundos, una curva "linear" movería un píxel en cada segundo, pero si damos "ease-in" los primeros segundos se moverá más lenta y luego continuaría de forma normal.

- Sintaxis:

`transition-timing-function: valor;`

- Linear. La transición tiene la misma velocidad en todo momento.
- ease. La transición empieza lenta, luego va rápida, y termina aun más lenta. (Por defecto)
- ease-in. La transición empieza lenta.
- ease-out. La transición termina lenta.
- ease-in-out. La transición empieza y termina lenta.
- cubic-bezier(d,d,d,d). Define tu propia curva de velocidad. Los valores posibles son de 0 a 1
  - `cubic-bezier(1,1,1,1)`
  - `cubic-bezier(0.25,0.1,0.25,1)`



# transition-delay

- Esta propiedad permite especifica un retardo (en segundos) para el comienzo de la transición

```
div {
 width: 100px;
 height: 100px;
 background: red;
 -webkit-transition: width 3s; /* Transición para Safari */
 -webkit-transition-delay: 1s; /* Retardo para Safari */
 transition: width 3s;
 transition-delay: 1s;
}

div:hover {
 width: 300px;
}
```

# Otras propiedades de transición

- Transition es la propiedad shorthand para definir la transición de un objeto.
- Las propiedades CSS3 de transición se pueden especificar una a una:

```
transition-property: width;
transition-duration: 2s;
transition-timing-function: linear;
transition-delay: 1s;
```

# Transición y transformación

- Las propiedades `transition` y `transform` se pueden combinar para conseguir efectos más vistosos.

```
div {
 width: 100px;
 height: 100px;
 background: red;
 -webkit-transition: width 2s, height 2s, -webkit-transform 2s; /*Safari*/
 transition: width 2s, height 2s, transform 2s;
}
```

```
div:hover {
 width: 300px;
 height: 300px;
 -webkit-transform: rotate(180deg); /* Safari */
 transform: rotate(180deg);
}
```



# Manos a la obra

Crear un nuevo documento HTML

## Formato del <div> original

Asignar un ancho de 80px, color de fondo #92B901, verdana blanco de 15px, padding superior e inferior de 20 px, radio-borde: 5px, opacidad 0,8

## Formato del <div> final

Ancho de 160px, color de fondo #1ec7e6, verdana blanco de 35px; padding superior e inferior de 40px, radio-borde:5px, opacidad: 1





*Animaciones*

- **Las animaciones CSS3** permiten animar la transición entre un estilo CSS y otro.
- Las animaciones constan de dos componentes:
  - Un estilo que describe la animación
  - Un conjunto de fotogramas que indican su estado inicial y final, así como posibles puntos intermedios en la misma.
- Las animaciones CSS tienen tres ventajas principales sobre las técnicas tradicionales de animación basada en scripts:
  - Es muy fácil crear animaciones sencillas sin tener que recurrir a Flash ni a JavaScript.
  - La animación se muestra correctamente, incluso en equipos poco potentes.
  - Al ser el navegador quien controle la secuencia de la animación, permitimos que optimice el rendimiento y eficiencia de la misma, por ejemplo, reduciendo la frecuencia de actualización de la animación ejecutándola en pestañas que no estén visibles.

# Configurar la animación

- Para crear una secuencia de animación necesitamos:
  - La propiedad `animation` que nos permitirá definir los tiempos de la animación (ritmo, duración, etc.)
  - La regla `@keyframes` (fotogramas) que nos permitirá definir la apariencia de la animación.

```
/* Fotogramas */
@keyframes prueba {
 from {background-color: green;}
 to {background-color: blue;}
}
```

```
/* Aplicamos la animación */
div {
 width: 100px;
 height: 100px;
 background-color: blue;
 color: white;
 animation-name: prueba;
 animation-duration: 4s;
}
```

Especifica el nombre de la regla  
`@keyframes` que se va a aplicar

Cantidad de tiempo que la animación dura  
la animación. Si no se especifica la  
propiedad la animación no tendría lugar.

```
div {
 width: 100px;
 height: 100px;
 background-color: blue;
 -webkit-animation-name: prueba; /* Chrome, Safari, Opera */
 -webkit-animation-duration: 4s; /* Chrome, Safari, Opera */
 animation-name: prueba;
 animation-duration: 4s;
}
/* Chrome, Safari, Opera */
@-webkit-keyframes example {
 from {background-color: blue;}
 to {background-color: green;}
}
/* Standard syntax */
@keyframes example {
 from {background-color: blue;}
 to {background-color: green;}
```



# @keyframes

- Los @keyframes son como fotogramas de la animación, los diferentes estados por los que va pasando la animación.
- Para crear la animación deberemos definir una lista de @keyframes. Esta lista de @keyframes indica en qué porcentaje a lo largo de la animación se alcanza un determinado estado (o fotograma).
- Toda lista de @keyframes debe contar al menos con dos estados: el inicial (estado de la animación cuando empieza, 0%) y el final (estado de la animación cuando termina, 100%).
- En el ejemplo visto antes, donde especificábamos las palabras "from" y "to", from representan el estado inicial y to el estado final (100%).

```
/* Fotogramas */
@keyframes prueba {
 0% {background-color: blue;}
 25% {background-color: orange;}
 50% {background-color: green;}
 100% {background-color: red;}
}
```

```
/* Aplicamos la animación a un div */
div {
 width: 100px;
 height: 100px;
 background-color: blue;
 animation-name: prueba;
 animation-duration: 4s;
}
```

```
div {
 width: 100px;
 height: 100px;
 background-color: red;
 -webkit-animation-name: prueba; /* Chrome, Safari, Opera */
 -webkit-animation-duration: 4s; /* Chrome, Safari, Opera */
 animation-name: prueba;
 animation-duration: 4s;
}
/* Chrome, Safari, Opera */
@-webkit-keyframes prueba{
 0% {background-color: red;}
 25% {background-color: yellow;}
 50% {background-color: blue;}
 100% {background-color: green;}
}
/* Standard syntax */
@keyframes prueba{
 0% {background-color: red;}
 25% {background-color: yellow;}
 50% {background-color: blue;}
 100% {background-color: green;}
}
```

Debe definirse otro bloque igual pero  
@-webkit-keyframes prueba

```
0 /* Fotogramas */
@keyframes prueba {
 0% {background-color: red; left:0px; top:0px;}
 25% {background-color: yellow; left:200px; top:0px;}
 50% {background-color: blue; left:200px; top:200px;}
 75% {background-color: green; left:0px; top:200px;}
 100% {background-color: red; left:0px; top:0px;}
}
```

```
/* Elemento a animar */
div {
 width: 100px;
 height: 100px;
 position: relative;
 background-color: red;
 animation-name: prueba;
 animation-duration: 4s;
}
```

# Otras propiedades

- `animation-delay`. Permite especificar un tiempo de retardo entre el momento en que el elemento se carga y el comienzo de la secuencia de la animación
- `animation-iteration-count`. Indicará el número de veces que se repite. Podemos indicar `infinite` para repetir la animación indefinidamente.
- `animation-direction`. Permite definir si la animación se va a ejecutar en orden normal (valor por defecto) en el orden inverso del definido o alternando.
  - Para definir que la animación debe de ser en el sentido contrario utilizaremos la palabra `reverse` y para indicar que se alterne, utilizaremos la palabra `alternate`.
- Al igual que con las transiciones, también podemos especificar la curva de velocidad de la animación. Esta propiedad es `animation-timing-function`, pudiendo elegir entre los mismos valores: `linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out` y `cubic-bezier(n,n,n,n)`



Imágenes

# Propiedades de imagen

- o Con la propiedad `border-radius` podemos redondear las esquinas de las imágenes

```
img {
 border-radius: 8px;
}

img {
 border-radius: 50%;
}
```

- o Con la propiedad `border` podemos poner un borde a la imagen, y con la propiedad `padding` podemos hacer que este borde no está pegado completamente a la imagen.

```
img {
 border: 1px solid #ddd;
 border-radius: 4px;
 padding: 5px;
}
```

- o Podemos poner una opacidad a la imagen con la propiedad `opacity`.

○ A una imagen, podemos ponerle un enlace e incluso añadir efectos para que al pasar el ratón por encima se ejecute alguna transición.

```
a {
 display: inline-block;
 border: 1px solid #ddd;
 border-radius: 4px;
 padding: 5px;
 transition: 0.3s;
}
```

```
a:hover {
 box-shadow: 0 0 2px 1px rgba
 (0, 140, 186, 0.5);
}
```

```



```



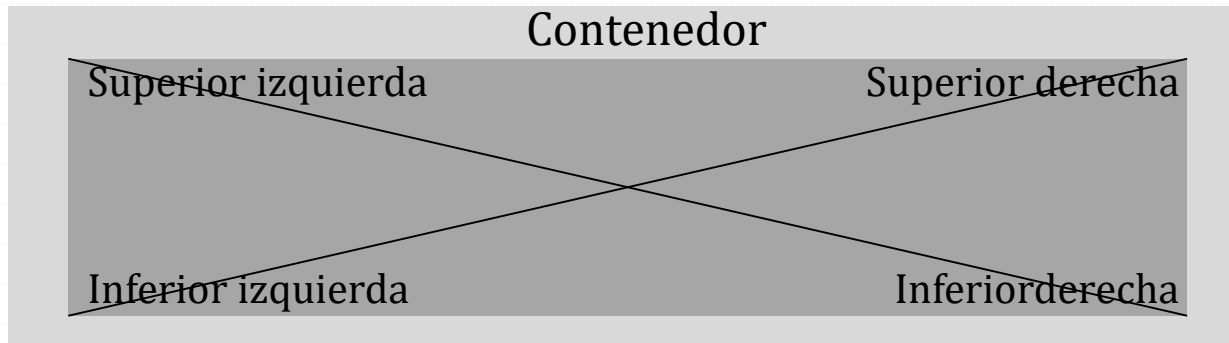
# Responsive images

- Las responsive images se ajustan automáticamente para encajar en el tamaño de la ventana
- Para conseguir que una imagen se escale según el tamaño de la ventana del navegador, incluiremos las siguientes líneas en el código CSS

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

# Texto en imágenes

Podemos añadir texto en las imágenes para que se superponga a él.



## HTML

```
<div class="contenedor">

 <div class="topleft">
 Texto izquierda
 </div>
</div>
```

## CSS

```
.contenedor {
 position: relative;
}
.topleft {
 position: absolute;
 top: 8px;
 left: 16px;
 font-size: 18px;
}
#foto {
 width: 100%;
 height: auto;
 opacity: 0.3;
}
```

# Filtros para imágenes

- Podemos añadir ciertos efectos visuales a una imagen con la propiedad `filter`.

```
img {
 -webkit-filter: grayscale(100%); /* Chrome, Safari, Opera */
 filter: grayscale(100%);
}

img {
 -webkit-filter: blur(4px); /* Chrome, Safari, Opera */
 filter: blur(4px);
}

img {
 -webkit-filter: hue-rotate(180deg); /* Chrome, Safari, Opera */
 filter: hue-rotate(180deg);
}

img {
 -webkit-filter: saturate(7); /* Chrome, Safari, Opera */
 filter: saturate(7);
}

img {
 -webkit-filter: sepia(100%); /* Chrome, Safari, Opera */
 filter: sepia(100%);
}
```