

Diseño de Interfaces Web

Transformaciones, transiciones y animaciones CSS3



Transformaciones, transiciones y animaciones CSS3



1. [Transformaciones](#)
2. [Transiciones](#)
3. [Animaciones](#)
4. [Clip-path](#)
- *. [Referencias](#)

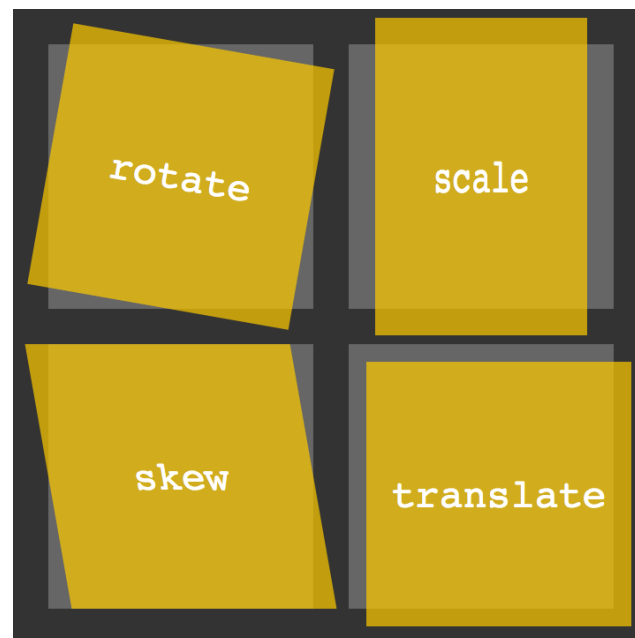
```
1 keyframes rocket-to-bottom-right {
2   0% {left:0%; top:0%;}
3   100% {left:100%; top:500px;}
4 }
5 @keyframes rocket-to-bottom-right {
6   0% {left:0%; top:0%;}
7   100% {left:100%; top:500px;}
8 }
9
10 .rocket-animation {
11   position: relative;
12   animation-name: rocket;
13   animation-duration: 5s;
14   animation-delay: 2s;
15   animation-count: infinite;
16   animation-iteration-count: alternate;
```



1. Transformaciones

- Transformaciones CSS

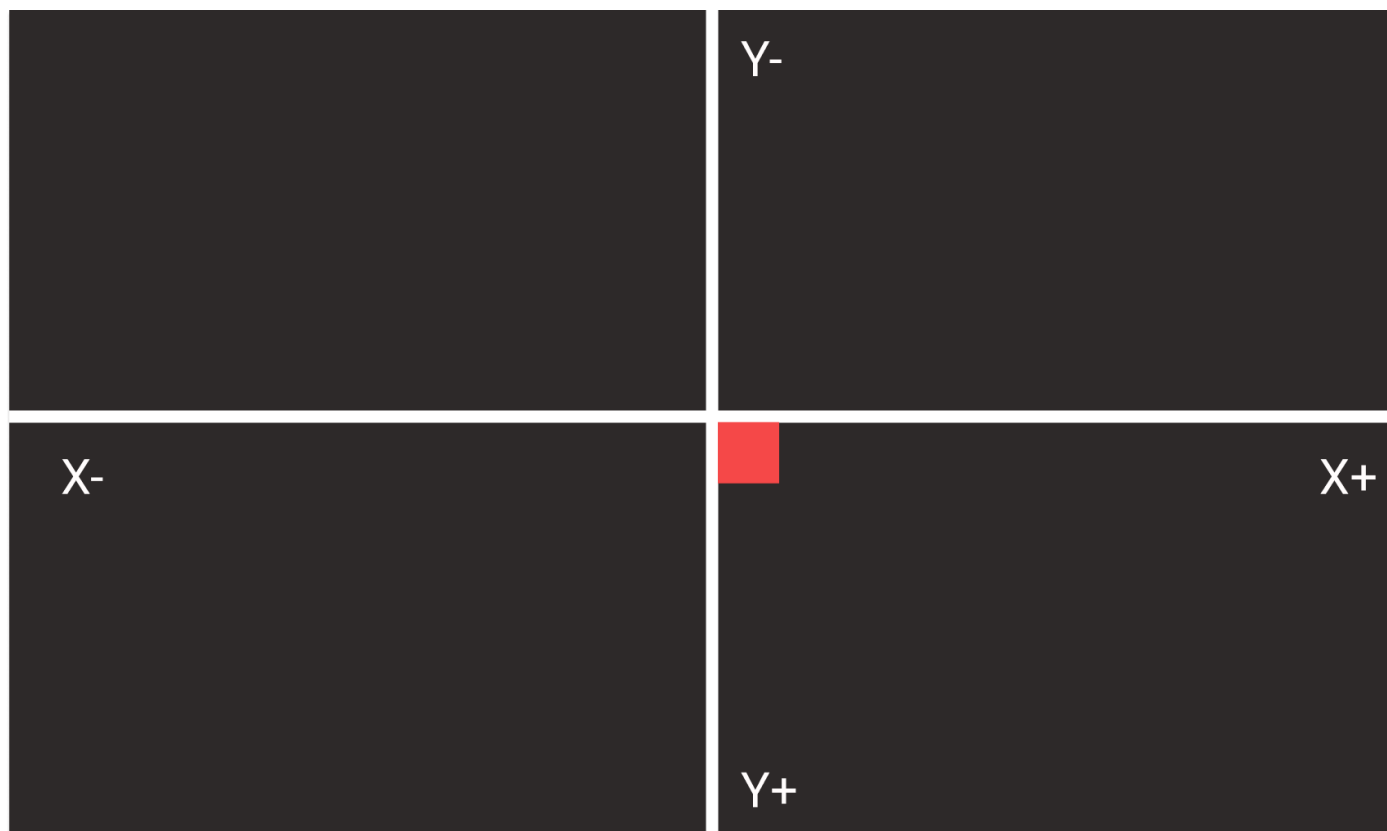
- Propiedad que permite modificar las coordenadas en el espacio de la **caja CSS** que representa al elemento
- Estas propiedades son:
 - ✓ Translate (translación)
 - ✓ Rotate (rotación)
 - ✓ Scale (escalado)
 - ✓ Skew (sesgado)





1. Transformaciones

- Eje coordenadas CSS





1. Transformaciones

- Translate

- Cambia de posición los elementos a izquierda, derecha, arriba o abajo (sin salir del flujo HTML)

```
selector {  
    transform: translateX(x);  
}
```

```
selector {  
    transform: translateY(y);  
}
```

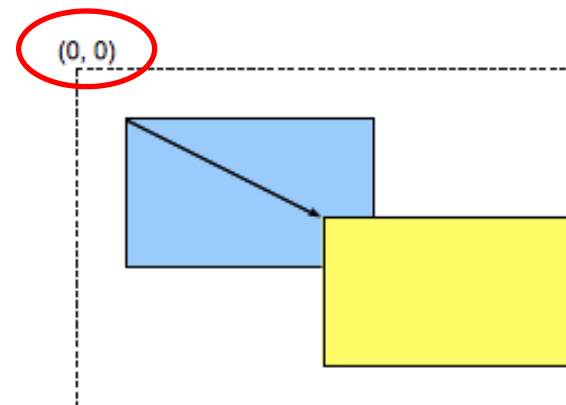
```
selector {  
    transform: translate(x,y);  
}
```



1. Transformaciones

- Translate

```
.translate {  
  transform: translate(50px, 100px);  
}
```





1. Transformaciones

- Scale
 - Cambia de tamaño los elementos

```
selector {  
    transform: scale(x,y);  
}
```

```
selector {  
    transform: scaleX(x);  
}
```

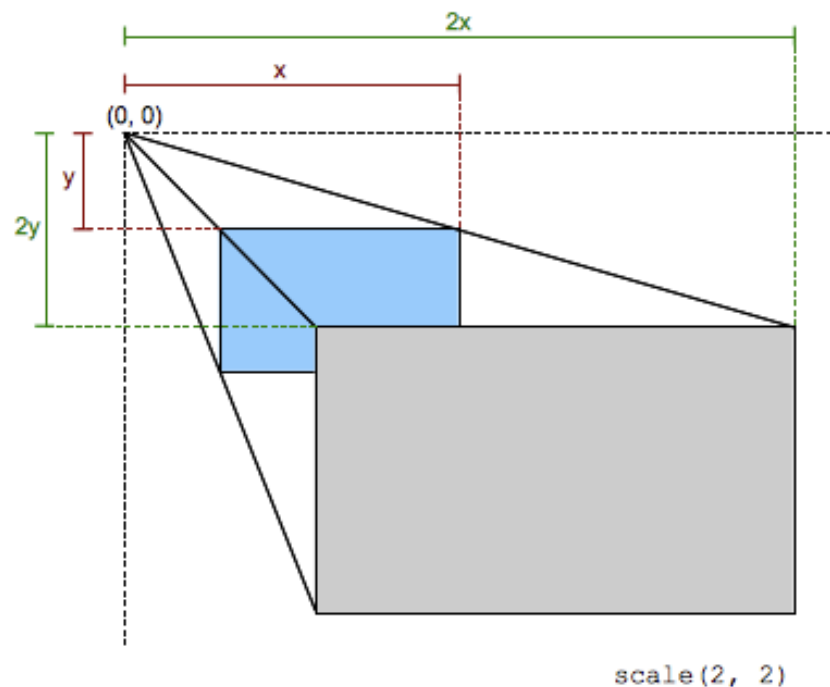
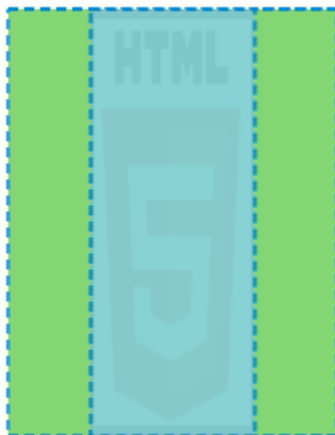
```
selector {  
    transform: scaleY(y);  
}
```



1. Transformaciones

- Scale

```
.scale {  
  transform: scaleX(0.5);  
}
```





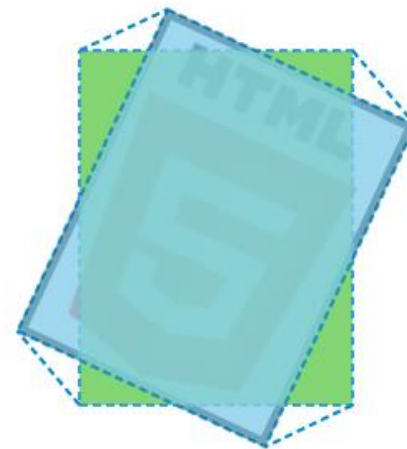
1. Transformaciones

- Rotate

- Gira los elementos un número determinado de grados en sentido horario
- El giro se puede expresar en grados (**deg**), radianes (**rad**), gradianes (**grad**), giros (**turn**)

```
/*  
    grados:    0-360 deg  
    radians:   0-6.28319 rad  
    gradians:  0-400 grad  
    turn:      0-1 turn  
*/
```

```
.rotate {  
    transform: rotate(25deg);  
}
```





1. Transformaciones

- Skew
 - Sesga o distorsiona los elementos con un cierto ángulo
 - Mismas unidades que rotate

```
selector {  
    transform: skew(anguloX, anguloY);  
}
```

```
selector {  
    transform: skewX(anguloX);  
}
```

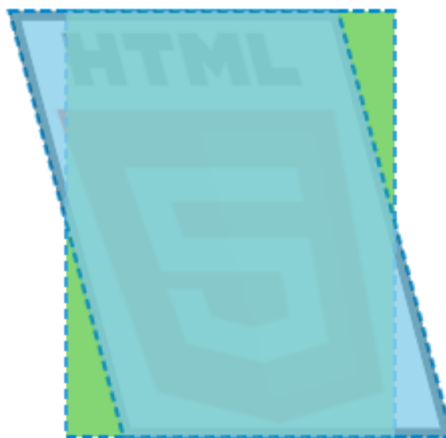
```
selector {  
    transform: skewY(anguloY);  
}
```



1. Transformaciones

- Skew

```
.skew {  
  transform: skewX(15deg);  
}
```

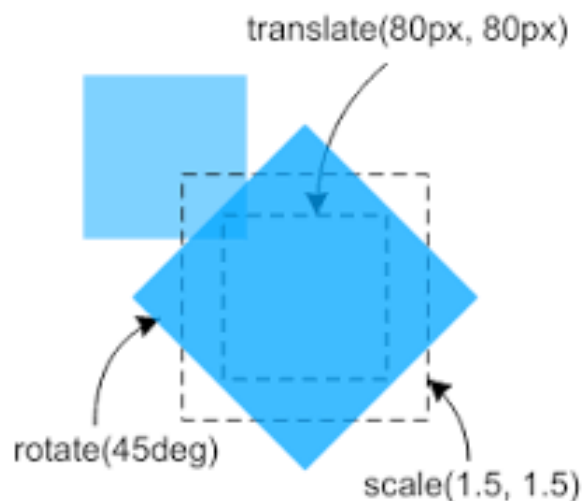




1. Transformaciones

- Combinar transformaciones
 - Es posible combinar varias transformaciones en una

```
selector {  
    transform: rotate(45deg) scale(1.5, 1.5)  
    translate(80px, 80px);  
}
```



Compound transform

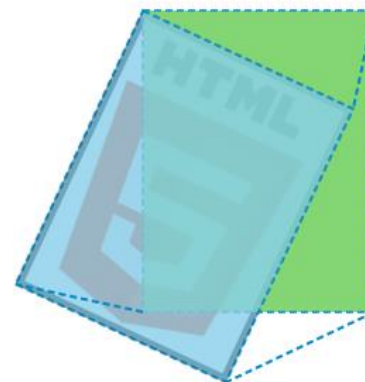


1. Transformaciones

- Origen de la transformación
 - Podemos cambiar el punto de referencia de la transformación
 - Por defecto es el centro de la caja (50%, 50%)
 - Se puede cambiar con **transform-origin**

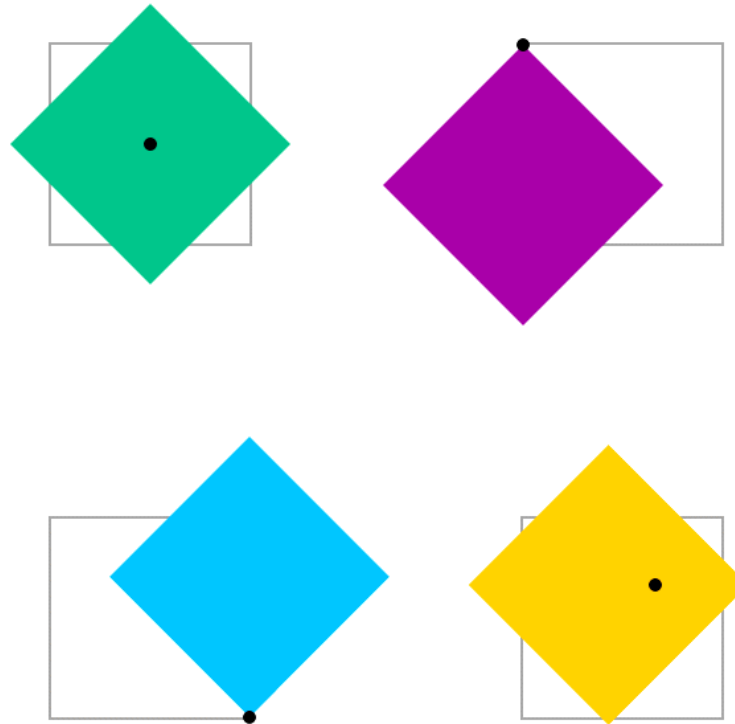
```
selector {  
    transform: rotate(45deg);  
    transform-origin: x, y;  
    /* top, bottom, center, left, right, %, px... */  
}
```

```
.cambio_origen {  
    transform: rotate(25deg);  
    transform-origin: top left;  
}
```



1. Transformaciones

- Origen de la transformación





2. Transiciones

- Transiciones CSS
 - Permiten cambiar valores de las propiedades de un elemento de forma gradual (produce una animación)
 - Estas propiedades pasarán de un *estado inicial* a un *estado final* que podremos visualizar
 - Necesitan un *trigger* o *disparador* para que la transición se ejecute:
 - pseudoclasas (**:hover**, **:active**, **:focus**,...)
 - mediante *JavaScript*



2. Transiciones

- Transiciones CSS

- A tener en cuenta:

- ✓ Los elementos **inline** no se pueden animar
 - ✓ Todo elemento tiene un **estado inicial** (si no lo indicamos, aplicará el estado inicial por defecto, lo que puede llevarnos a errores en la animación si no conocemos este valor)
 - ✓ No todas las propiedades son animables
 - ✓ Lista de propiedades animables:

https://www.w3schools.com/cssref/css_animatable.php



2. Transiciones

- Transiciones CSS
 - **transition** es un shorthand de las propiedades:
 - `transition-property`
 - `transition-duration`
 - `transition-timing-function`
 - `transition-delay`

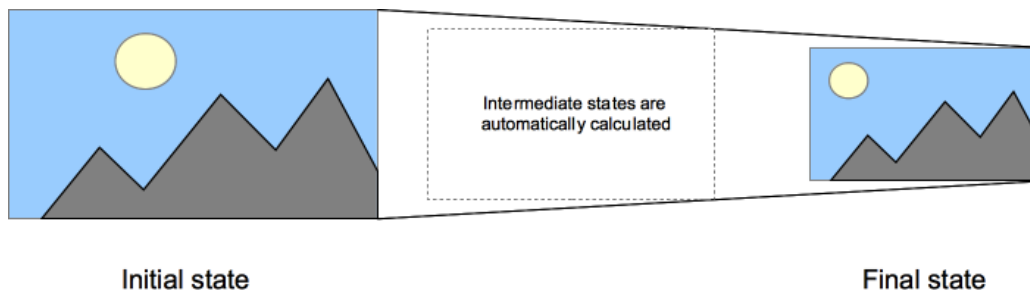


2. Transiciones

- Transiciones CSS

- Propiedades individuales

- `transition-property` → indica a qué propiedades se aplica la transición (**all** = todas)
- `transition-duration` → duración de la transición (s o ms)
- `transition-timing-function` → tipo de curva de aceleración de la animación (ease, ease-in, ease-out, ease-in-out, linear, cubic-bezier)
- `transition-delay` → retraso en la ejecución (s o ms)





2. Transiciones

- Transiciones CSS
 - Ejemplo:

```
h1 {  
  background-color: lightsteelblue;  
  border: 1px solid black;  
  color: white;  
  width: 400px;  
  margin: auto;  
  text-align: center;  
  transition-property: background-color, color;  
  transition-duration: 2000ms;  
  transition-timing-function: ease-in;  
}
```

```
h1:hover {  
  background-color: white;  
  color: darkslateblue;  
}
```



2. Transiciones

- Transiciones CSS
 - Ejecución de varias transiciones de forma simultánea

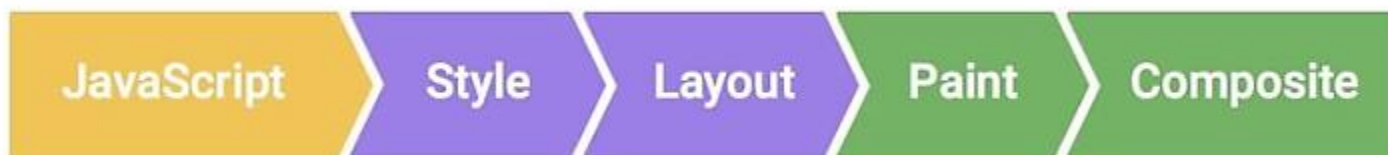
```
transition: transform 2s, color 4s, font-size 2s;
```

```
[ transition-property: transform, color, font-size;  
  transition-duration: 2s, 4s, 2s;
```



2. Transiciones

- Proceso de renderización de una página
 - Debemos tener en cuenta que las transiciones y las animaciones consumen recursos del sistema
 - Para optimizar estos recursos es necesario conocer la **renderización** de elementos que hace el navegador
 - Se compone de los siguientes pasos:



- Cada vez que se realizan transiciones o animaciones se pasa por este proceso

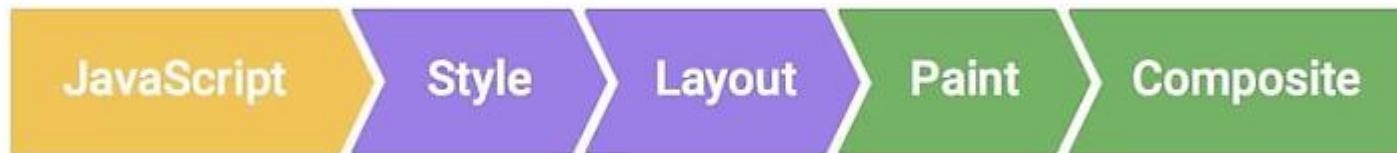


2. Transiciones

- Proceso de renderización

- Layout

- Tras la ejecución de los scripts y decidir las reglas CSS a aplicar, el navegador calcula la información geométrica de los elementos: su tamaño y ubicación en la página
 - Si modificamos alguna de las propiedades que cambian la geometría de los elementos (*width, height, position, top, left,...*) el navegador tendrá que verificar los otros elementos y redistribuirlos
 - Las áreas afectadas deberán volver a pintarse y los elementos pintados finales se volverán a componer en conjunto (se ejecutan *Layout, Paint y Composite*)





2. Transiciones

- Proceso de renderización

- Paint

- Proceso en el que se rellenan los píxeles
 - Es el caso de propiedades en las que se dibuja texto, colores, imágenes, bordes y sombras
 - Si cambiamos una de estas propiedades, se omite el diseño (layout) pero se ejecutan *Paint* y *Composite*





2. Transiciones

- Proceso de renderización
 - Composite
 - Se encarga de dar el orden adecuado a las capas. Las combina y las dibuja
 - Si modificamos una propiedad en la que no se requiere diseño ni pintura (propiedades *transform* u *opacity*), el navegador omite estas fases y continúa con la composición de la página
 - Proceso que menos recursos consume





2. Transiciones

- Proceso de renderización
 - Recomendaciones
 - Procurar usar propiedades que solo ejecuten composite: **transform** y **opacity**
 - Evitar usar animaciones simultáneas



3. Animaciones

- Animaciones CSS
 - Variante de las transiciones: permiten definir **varios estados intermedios** entre un estado inicial y un estado final
 - **No necesitan un disparador** para ejecutarse
 - Se pueden ejecutar cierto número de veces (o infinitas)
 - Se crean definiendo reglas (*at-rules*) **@keyframes**

```
@keyframes cambia_color {  
    from {background-color: green;}  
    25% {background-color: yellow;}  
    50% {background-color: blue;}  
    75% {background-color: white;}  
    to {background-color: red;}  
}
```



3. Animaciones

- Animaciones CSS
 - **animation** es un shorthand de:
 - ✓ `animation-name`
 - ✓ `animation-duration`
 - ✓ `animation-timing-function`
 - ✓ `animation-delay`
 - ✓ `animation-iteration-count`
 - ✓ `animation-direction`
 - ✓ `animation-fill-mode`
 - ✓ `animation-play-state`



3. Animaciones

- Propiedades animaciones
 - `animation-name` → referencia a una animación definida previamente en un **keyframe**
 - `animation-duration` → duración en segundos o ms
 - `animation-delay` → retardo para el inicio
 - `animation-timing-function` → tipo de curva de aceleración de la animación (ease, ease-in, ease-out, ease-in-out, linear, cubic-bezier)



3. Animaciones

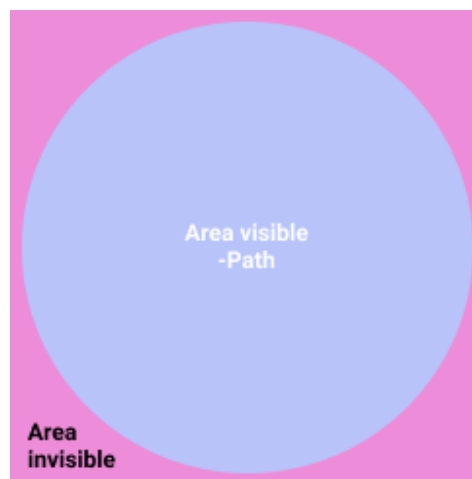
- Propiedades animaciones

- `animation-iteration-count` → número de veces que se ejecuta la animación (puede ser infinito)
- `animation-direction` → desde qué punto se inicia la animación (normal, reverse, alternate, alternate-reverse)
- `animation-fill-mode` → si los estilos de la animación se aplican después de que termine (backwards, normal, forwards, both)
- `animation-play-state` → para pausar y reanudar la animación (running, paused)



4. Clip-path

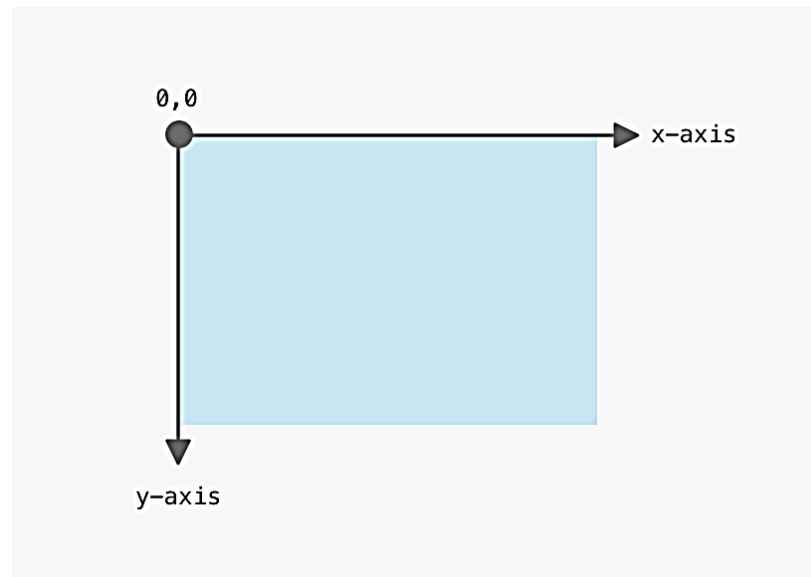
- Propiedad clip-path
 - Permite realizar un recorte sobre un elemento con una forma concreta (círculo, polígono, elipse, ...)
 - La región externa queda oculta y no seleccionable
 - El área visible del recorte se denomina **path**
 - El recorte puede ser especificado por un método de figuras o por una url





4. Clip-path

- Coordenadas
 - Utiliza el mismo sistema de coordenadas visto hasta ahora
 - Son relativas al elemento, es decir la posición inicial **0,0** es la parte **superior izquierda** del elemento

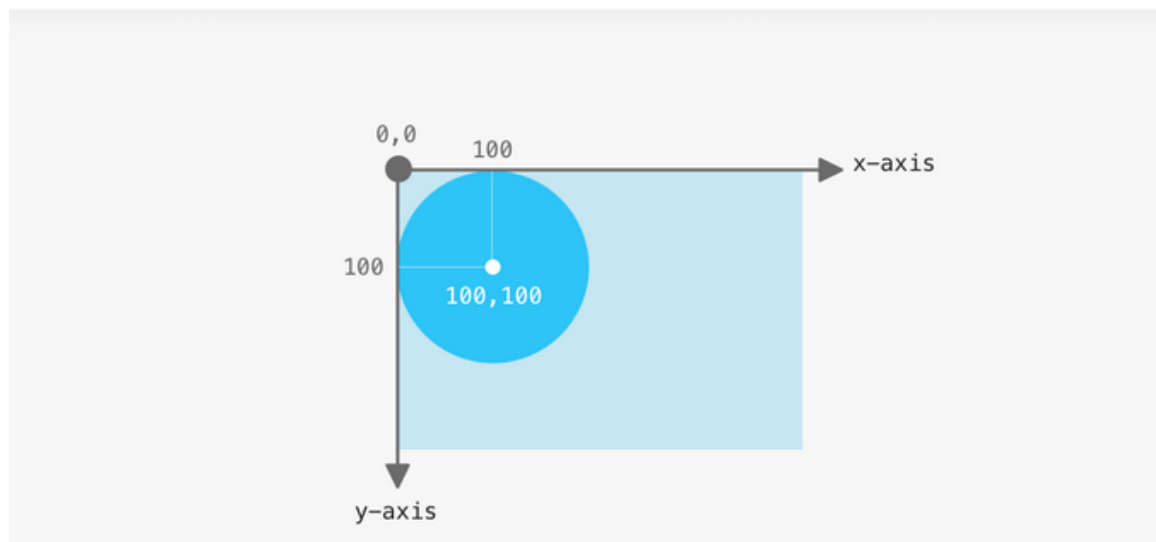




4. Clip-path

- Valores
 - circle()

```
.card {  
  background-color: #77cce9;  
  clip-path: circle(100px at 100px 100px);  
}
```

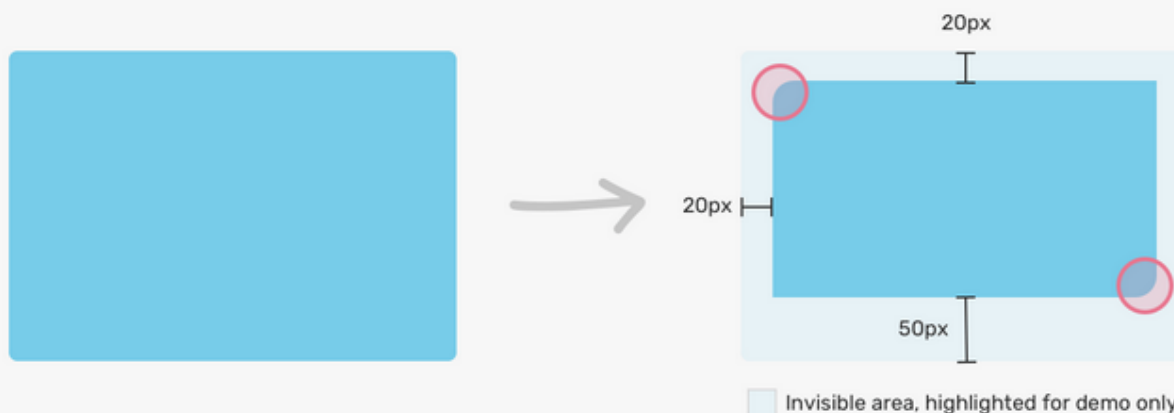




4. Clip-path

- Valores
 - inset()

```
.card {  
  background-color: #77cce9;  
  clip-path: inset(20px 20px 50px round 15px 0);  
}
```

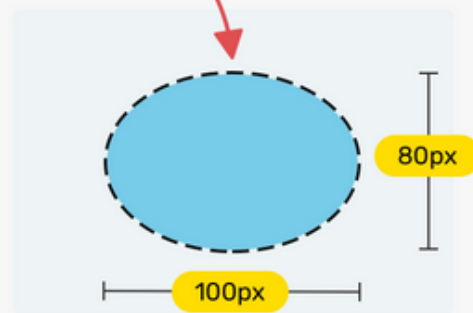
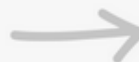
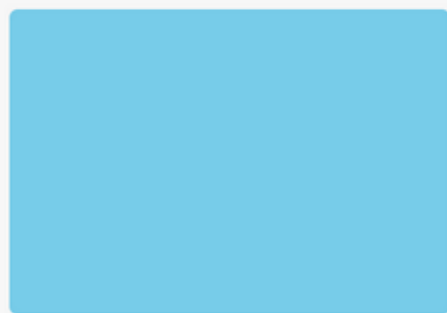




4. Clip-path

- Valores
 - ellipse()

```
.card {  
  background-color: #77cce9;  
  clip-path: ellipse(100px 80px at center);  
}
```

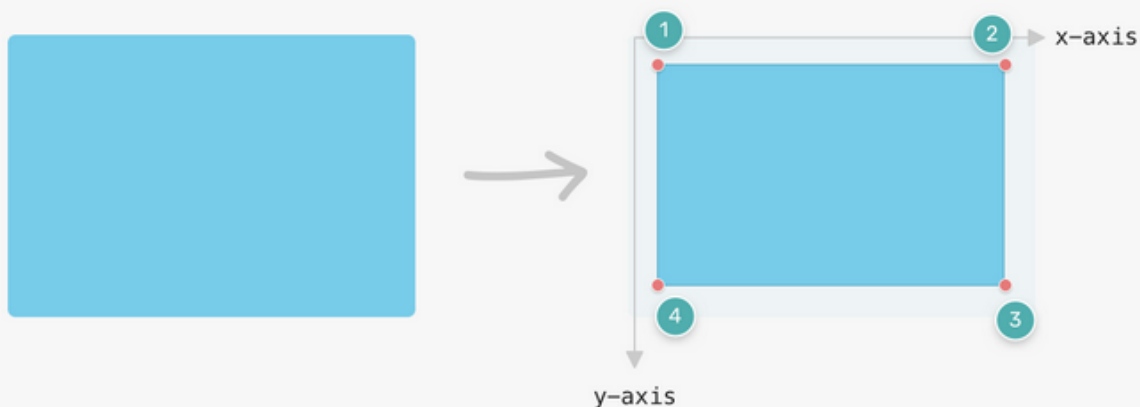




4. Clip-path

- Valores
 - polygon()

```
.card {  
  background-color: #77cce9;  
  clip-path: polygon(5% 5%, 95% 5%, 95% 95%, 5% 95%);  
}
```





4. Clip-path

- Valores
 - path()

HTML

```
<svg class="svg">
  <clipPath id="triangle">
    <path d="M0.05,0.05 h1 v1"></path>
  </clipPath>
</svg>
```

CSS

```
.card {
  background-color: #77cce9;
  clip-path: url(#triangle);
}
```

The diagram illustrates the application of a clip-path. On the left, a solid blue square represents the initial state. A grey arrow points to the right, where the result is shown: a light blue square with a darker blue right-angled triangle in the top-left corner. A red arrow originates from the 'url(#triangle)' property in the CSS code and points directly to the triangle shape in the final result, indicating that the clip-path is used to restrict the background color to the area defined by the triangle.

4. Clip-path



- CSS clip-path maker
 - <https://bennettfeely.com/clippy/>

CSS clip-path maker [Tweet](#)

clip-path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%, 50% 70%, 21% 91%, 32% 57%, 2% 35%, 39% 35%);



*. Referencias

- Bibliografía y referencias
 - Libro “Diseño de Interfaces Web” de Eugenia Pérez Martínez / Pello Xabier Altadill Izura – Ed. Garceta
 - Libro “Diseño de Interfaces Web” de Diana García-Miguel López – Ed. Síntesis
 - Taller de Openwebinars.net “Transiciones, transformaciones y animaciones”, de Juan Diego Pérez
 - MDN Web Docs: <https://developer.mozilla.org/es/>
 - <https://www.w3schools.com/>
 - <https://ishadeed.com/article/clip-path/>