CSS 3

octubre 2018

Prof. Ainhoa Iglesias

ÍNDICE:

CSS 3

PROPIEDADES CSS 3 BÁSICAS

1. BORDES REDONDEADOS

Podemos cambiar la apariencia de nuestros bordes, de cuadrados a redondeados.

Se puede configurar el radio de la curvatura con el mismo valor para todos los lados del borde, o bien configurar uno diferente para cada lado, de manera individual[1]:

```
border-radius

border-top-left-radius
border-top-right-radius
border-bottom-right-radius
border-bottom-left-radius
```

También podemos unificar las propiedades (en el sentido de las agujas de reloj):

```
border-radius: 10px 20px 30px 60px;
```

Propiedades temporales:

```
-webkit-border-radius

-webkit-border-top-left-radius
-webkit-border-top-right-radius
-webkit-border-bottom-right-radius
-webkit-border-bottom-left-radius

-moz-border-radius
-moz-border-radius-topleft
-moz-border-radius-topright
-moz-border-radius-bottomright
-moz-border-radius-bottomleft
```

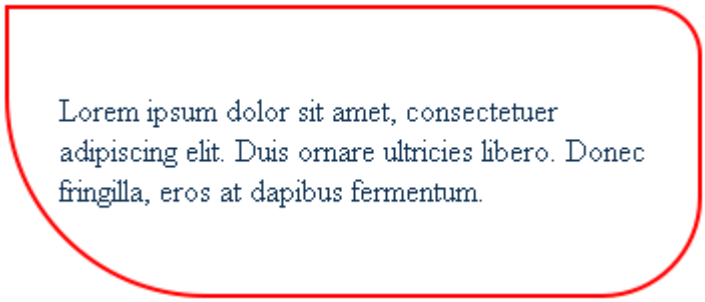
Ejemplo 1:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis ornare ultricies libero. Donec fringilla, eros at dapibus fermentum.

[1] http://www.the-art-of-web.com/css/border-radius/

```
#esquinas-redondeadas{
    border:2px solid #F00;

    border-radius:6px;
}
```

Ejemplo 2:



Forma unificada:

```
#esquinas-redondeadas-desiguales{
    border:2px solid #F00;

    border- radius: 0px 25px 50px 100px;
}
```

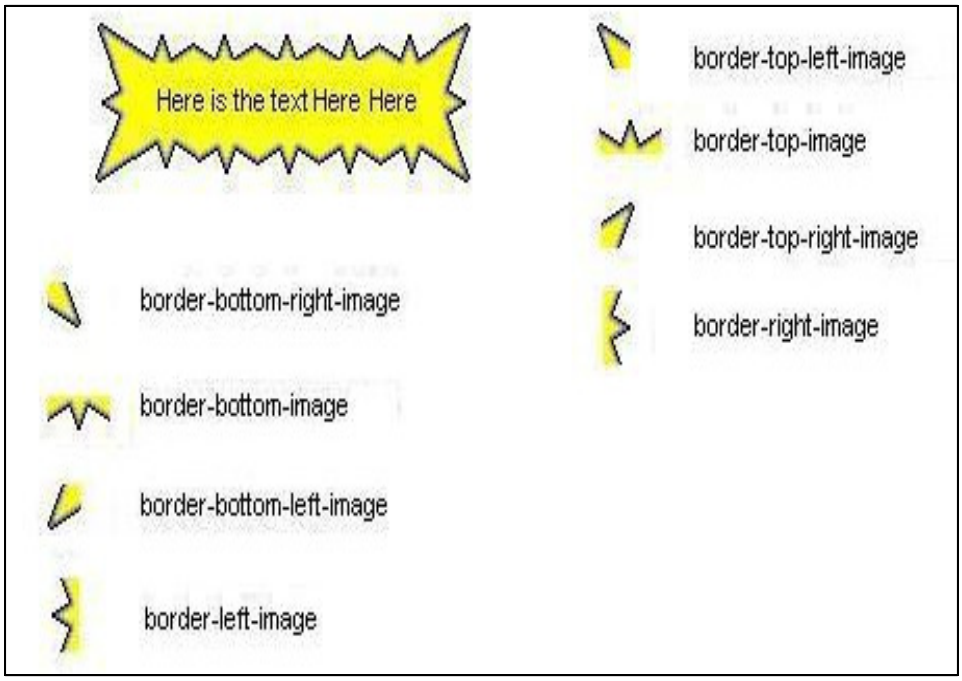
O la forma compleja:

```
#esquinas-redondeadas-desiguales{
    border:2px solid #F00;

    border-top-left-radius: 0px;
    border-top-right-radius: 25px;
    border-bottom-right-radius: 50px;
    border-bottom-left-radius: 100px;
}
```

2. IMÁGENES EN BORDES

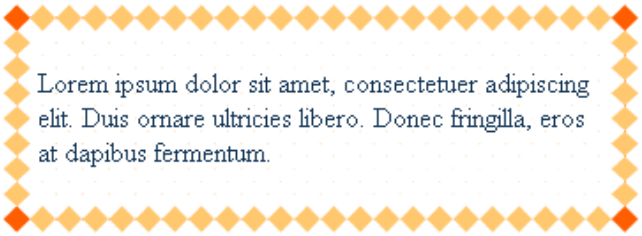
Podemos utilizar imágenes para crear el borde de un elemento

```
border-image
    border-top-image
    border-right-image
    border-bottom-image
    border-left-image


border-corner-image
    border-top-left-image
    border-top-right-image
    border-bottom-left-image
    border-bottom-right-image
```



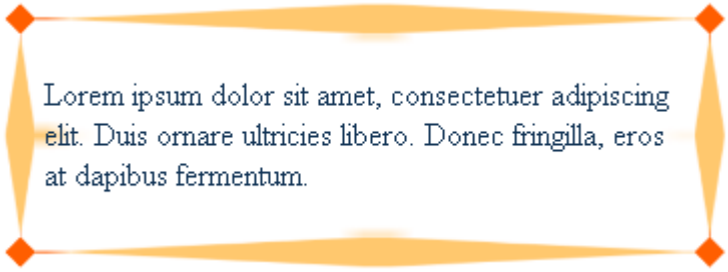
Y decidir si queremos que se repita la imagen o bien se estire, mediante las propiedades `round` y `stretch`.

Ejemplo 1:




```
#borde-imagen{
    border:15px solid orange;
    border-image: url(border.png) 27 27 27 27 round round;
}
```

Ejemplo 2:




```
#borde-imagen2{
    border:15px solid orange;

    border-image: url(border.png) 27 27 27 27 stretch stretch;
}
```

También podemos utilizar generadores[2] automáticos.

[2] http://border-image.com/

3. AJUSTES DE LOS BORDES

Podemos configurar el modo en que se repetirá la imagen de los bordes:

```
border-fit
    border-fit-length
    border-fit-width
```

`border-fit-length` determina la longitud de la imagen en los 4 lados:

```
border-fit-length
    border-top-fit-length
    border-right-fit-length
    border-bottom-fit-length
    border-left-fit-length
```

mientras que `border-fit-width` determina la anchura:

```
border-fit-width
    border-top-fit-width
    border-right-fit-width
    border-bottom-fit-width
    border-left-fit-width
```

Los valores de estas propiedades pueden ser: `clip`, `repeat`, `scale`, `stretch`, `overwrite`, `overflow`, `space`

También tenemos la propiedad `border-corner-fit` que determina la configuración de las imágenes en las esquinas.

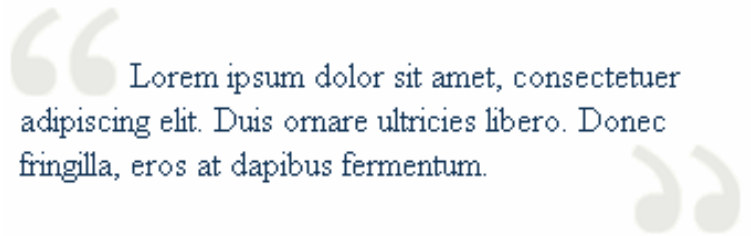
4. MÚLTIPLES IMÁGENES EN EL FONDO

Hasta ahora, sólo podíamos utilizar una única imagen como fondo de un elemento.

Pero con CSS 3, ya podemos utilizar varias imágenes como fondo del mismo elemento:

```
background:
    url(img) top left no-repeat,
    url(img) top right no-repeat,
    url(img) bottom left no-repeat,
    url(img) bottom right no-repeat;
```

Ejemplo:

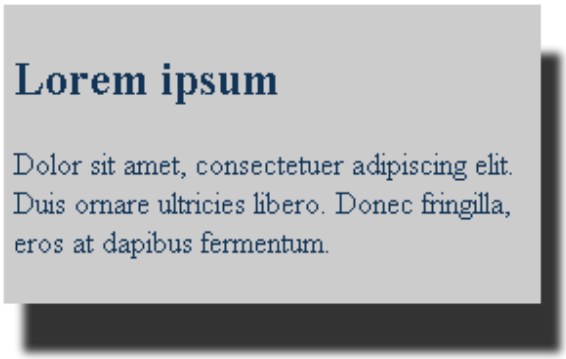



```
#multi-fondo{
    background: url(quote_open.gif) no-repeat top left,
url(quote_close.gif) no-repeat bottom right;
}
```

5. SOMBRAS EN BLOQUES

Para aplicar una sombra a un bloque, configuraremos el color, longitud horizontal y vertical, así como el radio de desenfoque de la sombra.

Ejemplo:



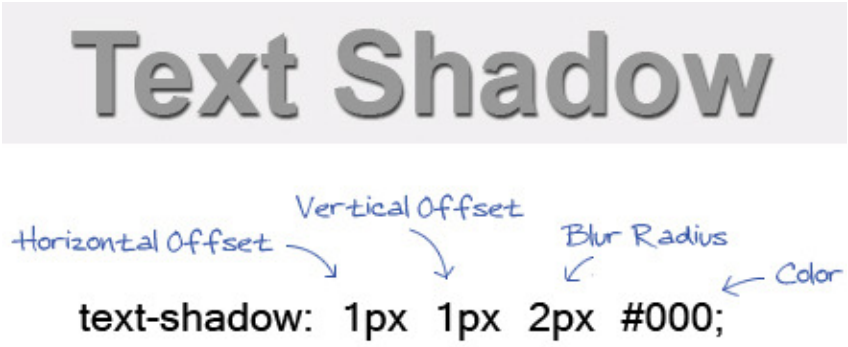

```
#bloque-sombra{
    box-shadow:10px 25px 5px #333;
}
```

Podemos usar generadores[3] automáticos[4]para que nos sea más cómoda la implementación.

[3] http://www.layerstyles.org/builder.html
[4] http://nicolasgallagher.com/css-drop-shadows-without-images/demo/

6. SOMBRAS EN TEXTOS

De la misma forma que podemos hacer sombras en los bloques, también vamos a poder aplicárselas a los textos:



También podremos aplicar varias sombras al mismo elemento

Ejemplo 1:


```
h2{
     text-shadow: 1px 5px 2px #F00;
}
```

Ejemplo 2:


```
h2{
     text-shadow: 0 1px 0 #fff, 0 -1px 0 #000;
}
```

También podemos usar un Generador Automático[5]

[5] http://westciv.com/tools/shadows/

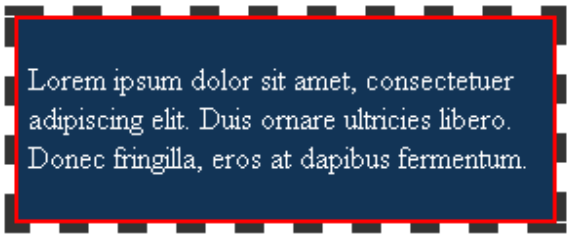
7. CONTORNO DE BLOQUES

El contorno de un elemento es similar al borde, con la diferencia de que no interfiere en el modelo de caja.

Por defecto, el contorno empieza justo por fuera del límite del borde, pero se puede ajustar a más distancia:

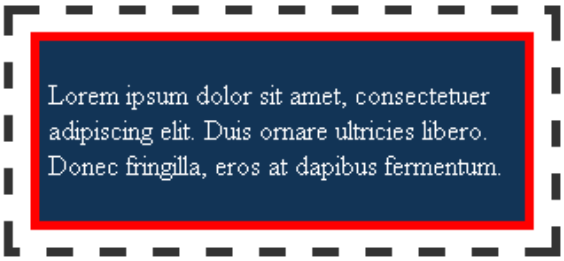
```
outline:
     outline-color:
     outline-style:
     outline-width:

outline-offset:
```

Ejemplo 1:




```
#outline{
     border:2px solid #F00;
     outline:5px dashed #333;
}
```

Ejemplo 2:




```
#outline2{
     border:2px solid #F00;
     outline:5px dashed #333;
     outline-offset: 10px;
}
```

8. TIPOGRAFÍAS PERSONALIZADAS

A partir de ahora vamos a poder utilizar la tipografía[6] que queramos en nuestras páginas (aunque el usuario no la tenga instalada en su dispositivo).

Para ello, primero tendremos que declarar la tipografía que vamos a utilizar, indicar dónde se encuentra y el formato de dicha fuente.

Cada navegador tiene soporte para diferentes formatos[7]:

```
@font-face {
  font-family: 'MyWebFont';
  src: url('webfont.eot'); /* IE9 Compat Modes */
  src: url('webfont.eot?#iefix') format('embedded-opentype'), /* IE6-IE8 */
  url('webfont.woff') format('woff'), /* Modern Browsers */
  url('webfont.ttf')  format('truetype'), /* Safari, Android, iOS */
  url('webfont.svg#svgFontName') format('svg'); /* Legacy iOS */
}
```

Y después, ya estamos listos para utilizarla en cualquier elemento (como si fuera una tipografía común):

```
h1{
     font-family: 'Adventpro', Helvetica, Sans-Serif;  }
```

Jugando con CSS 3

También podemos utilizar un generador automático[8], sobre todo para la compatibilidad entre diferentes navegadores[9].

Nota:

Cuidado con las licencias de las tipografías.

Antes de incluir una tipografía (que cualquier usuario va a poder descargar y usar posteriormente en su dispositivo), tendremos que asegurarnos de que su licencia nos permite incrustarla en una página web.

Repositorio de Fuentes de uso gratuito en web

Google Font Directory[10].

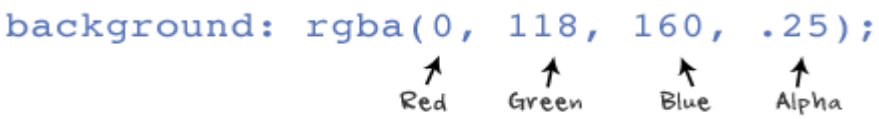
Repositorio de Fuentes de pago

Adobe Typekit[11]

[6] http://perishablepress.com/press/2010/04/14/visual-walkthrough-font-face-css/
[7] https://ksesocss.blogspot.com/2012/05/font-face-y-sus-problemas-guia-de-uso-y.html
[8] https://www.fontsquirrel.com/tools/webfont-generator
[9] http://webdesignerwall.com/general/font-face-solutions-suggestions
[10] https://fonts.google.com/
[11] https://typekit.com/

9. RGBA

Además de indicar el color en formato RGB (en vez de hexadecimal), también podremos modificar la opacidad del elemento:



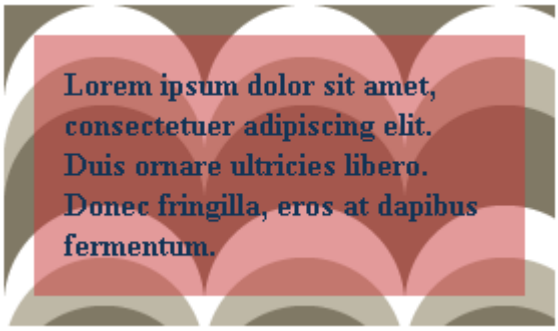
Los tres primeros valores indican la cantidad de Rojo, Verde y Azul de nuestro color.

El 4º valor, Alpha, indica el nivel de transparencia. Debe estar entre 0 y 1 (0 = transparente / 1 = opaco).

El color RGBA se puede utilizar en fondos, colores de primer plano, bordes, etc.

También podemos modificar la opacidad de un elemento, aunque definamos el color en hexadecimal. Bastaría con utilizar la propiedad `opacity`.

Ejemplo 1:




```
#bloque-transparencia-rgb{
    background:#FFF url(fondo2.gif);
}

#bloque-transparencia-rgb p{
    margin:10px;
    padding: 15px;
    font-weight:bold;
    background: rgba(200, 54, 54, 0.5);
}
```

Ejemplo 2:


```
#bloque-transparencia-rgb-bordes{
    background:#FFF url(fondo2.gif);
}

#bloque-transparencia-rgb-bordes p{
    margin:10px;
    padding: 15px;
    font-weight:bold;
    border:20px solid rgba(200, 54, 54, 0.5);
}
```

Ejemplo 3:


```
#bloque-transparencia{
    background:#FFF url(fondo.gif);
}

#bloque-transparencia p{
    color:#333;
    opacity: 0.8;
}
```

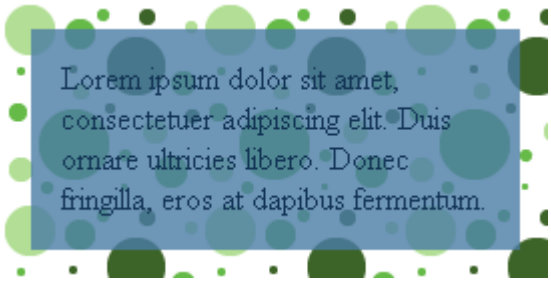
10. HSLA

Otra manera de indicar el color es mediante las propiedades de Tono (Hue), Saturación (Saturation) y Luminosidad (Lightness).

También se puede indicar la opacidad (Alpha), en el último valor:

```
background: hsla(H,S,L,A);
```

Ejemplo:




```
#bloque-hsla{
    background:#FFF url(fondo.gif);
}

#bloque-hsla p {
    margin:10px;
    padding: 15px;
    background: hsla(207,38%,47%,.8);
}
```

11. COLUMNAS MÚLTIPLES

A partir de ahora, podemos visualizar un texto en columnas[12].

Además del nº de columnas que queramos configurar, podemos indicar la separación entre ellas, dibujar una línea en esa separación y hasta el tamaño de cada columna:

```
column-count:
column-width:
column-gap:
column-rule:
```

Ejemplo:

Lorem ipsum dolor sit amet, augue sed dui vehicula posuere. Nullam sed mauris et sodales. Praesent vestibulum,

```
#bloque-columnas{
    column-count: 3;
    column-gap: 20px;
    column-rule: 1px solid #00000;
}
```

[12] http://www.alistapart.com/articles/css3multicolumn

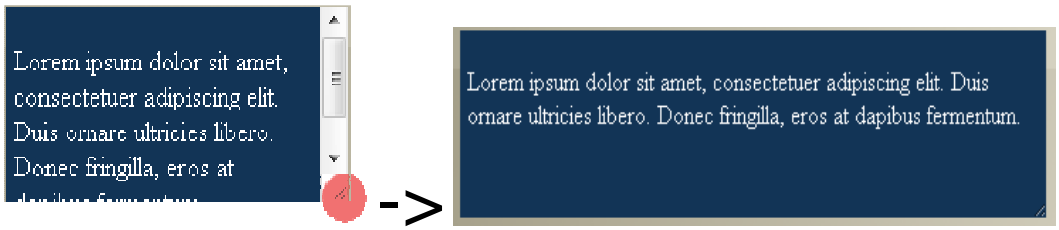
12. REDIMENSIONAMIENTO DE BLOQUE

Para aquellos bloques donde aparezca una barra de desplazamiento, podemos configurar la posibilidad de que el usuario redimensione el bloque según sus necesidades:

```
resize: both / horizontal / vertical;
```

Es aconsejable implementar una altura y anchura máximas, para que el redimensionamiento del usuario tenga unos límites.

Ejemplo:




```
#resize{
     width:200px; height:100px;
     overflow:auto;
     resize: both;
}
```

13. GRADIENTES

Podemos generar fondos con gradientes, sin necesidad de utilizar imágenes de fondo.

Podremos configurar múltiples[13] combinaciones[14] de colores, tamaños, direcciones...

-webkit-gradient(linear, 0% 0%, 0% 100%, from(#eee), to(#999));
-moz-linear-gradient(100% 100% 90deg, #999, #eee);

Webkit:

```
-webkit-gradient(<type>, <point> [, <radius>]?, <point> [,
<radius>]? [, <stop>]*)
```

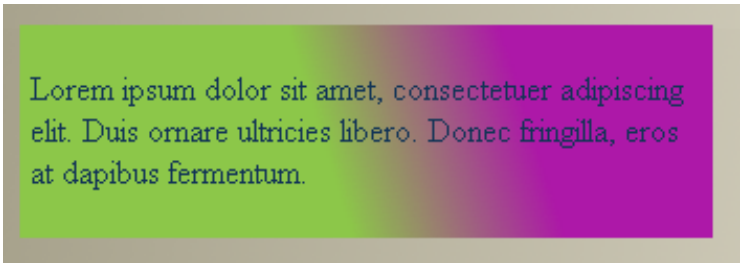
Mozilla:

```
-moz-linear-gradient( [<point> || <angle>,]? <stop>, <stop> [,
<stop>]* )
```

Ejemplo 1:

```
background: -webkit-gradient(linear, 0 0, 0 100%, from(red),
to(blue));
background: -moz-linear-gradient(top, red, blue);
```

Ejemplo 2:




```
#gradiente{
```

[13] http://www.the-art-of-web.com/css/linear-gradients/
[14] http://www.the-art-of-web.com/css/radial-gradients/

```
background: -webkit-gradient(linear,left bottom,right top,color-
stop(0.44, rgb(140,199,73)),color-stop(0.72, rgb(173,24,168)));

background:-moz-linear-gradient(left bottom,rgb(140,199,73)
44%,rgb(173,24,168) 72%);
}
```

También podemos utilizar varios generadores[15] automáticos[16], muy útiles para que sean compatibles con todos los navegadores que tienen soporte[17] y no sólo generar gradientes, al uso: podemos crear patrones[18] e incluso generar el patrón partiendo de una imagen concreta[19].

[15] http://colorzilla.com/gradient-editor/
[16] http://westciv.com/tools/gradients/
[17] http://lea.verou.me/demos/cssgradientsplease/
[18] http://lea.verou.me/css3patterns/
[19] http://gradient-scanner.com/

14. FILTROS

Los filtros nos permiten manipular tanto elementos html como imágenes en varios aspectos. La sintaxis base es[20] [21]:

```
filter: filter(value);
```

- Desenfoque: `blur(px)`
- Brillo: `brightness(valor)` – de 0 a 1
- Saturación: `saturate(%)`
- Tono girado: `hue-rotate(deg)`
- Contraste: `contrast(%)` -más de 100% añade contraste-
- Invertir: `invert(%)`
- Escala de grises: `grayscale(%)`
- Sepia: `sepia(%)`
- Opacidad: `opacity(%)`
- Sombra: `drop-shadow(shadow)`

SVG: Usamos un elemento que contenga el estilo con el svg que queramos utilizar como filtro: `<filter id=”contenedor-svg”>...</filter>`

Después aplicamos ese elemento al elemento que queremos aplicar el filtro:

```
div {filter: url(#contenedor-svg); }
```

Además podremos combinar varios filtros a la vez:

```
div {filter: grayscale(100%) sepia(100%); }
```

Se puede utilizar un generado automático[22] [23].

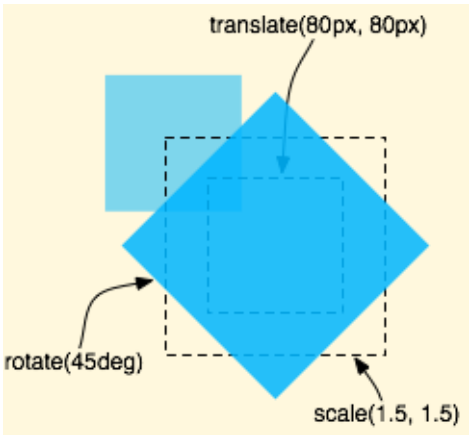
[20] https://css-tricks.com/almanac/properties/f/filter/
[21] http://css3.bradshawenterprises.com/filters/
[22] http://html5-demos.appspot.com/static/css/filters/index.html
[23] http://bennettfeely.com/filters/

15. TRANSFORMACIONES 2D

Podemos usar diferentes transformaciones sobre un elemento (bloque y/o texto): rotaciones, escalado, desplazamiento y sesgado [24] [25].

Lo mejor es que podemos combinarlas entre sí:

```
transform: rotate(deg);
transform: translate(x, y);
transform: scale(x, y);
transform: skew(deg, deg);
```



Ejemplo 1:

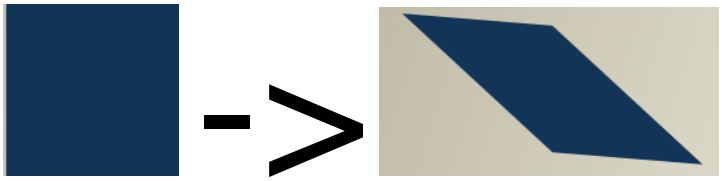

```
#menu li{
    transform: rotate(-5deg);
}
```

Ejemplo 2:

[24] https://css-tricks.com/almanac/properties/t/transform/
[25] http://css3.bradshawenterprises.com/transforms/

```
#menu li{
    transform: rotate(-5deg);
}

#menu li a{
    transform: rotate(5deg);
}
```

Ejemplo 3:



```
#transformacion2{
    transform: skew(45deg, 5deg);
}
```

Podemos utilizar generadores automáticos [26] [27]

MATRIX

Sirve para combinar varias transformaciones de una forma más eficaz, controlando los valores de forma matemática[28]:

```
#transformacion{
    transform: matrix(4.922, -0.944, 0.589, 3.944, 44.889,
70.000);
}
```

Podemos utilizar generadores automáticos[29] [30]

Iconos, formas

Estas transformaciones nos permiten simular imágenes[31] sencillas[32], simplemente con código css[33] y hacer máscaras[34] [35] .

[26] http://westciv.com/tools/transforms/
[27] http://www.hitthebits.com/2012/06/css3-3d-transform-playground.html
[28] http://www.useragentman.com/blog/2011/01/07/css3-matrix-transform-for-the-mathematically-challenged/
[29] http://www.useragentman.com/matrix/
[30] http://meyerweb.com/eric/tools/matrix/
[31] http://1stwebdesigner.com/css-shapes/
[32] http://nicolasgallagher.com/pure-css-gui-icons/demo/
[33] http://www.imgtocss.com/
[34] https://css-tricks.com/clipping-masking-css/
[35] http://codepen.io/CreativeJuiz/pen/Hizkh

16. TRANSFORMACIONES 3D

La mayor parte de las transformaciones 2D tiene su versión 3D.

```
translate3d(x, y, z)
translateZ(z)

scale3d(sx, sy, sz)
scaleZ(sz)

rotateX(value)
rotateY(value)
rotate3d(x, y, z)

matrix3d(…)

perspective(value)
```

Para el caso de una matriz, los valores aumentan por lo que lo lógico es utilizar un generador automático (como los indicados en el punto anterior).

Algunos ejemplos aclaratorios: [36] [37] [38]

[36] https://www.webkit.org/blog/386/3d-transforms/
[37] http://desandro.github.io/3dtransforms/
[38] http://css-tricks.com/almanac/properties/p/perspective/

17. TRANSICIONES

Las transiciones nos permiten cambiar propiedades CSS de un elemento durante una duración determinada: cambio de color, de posición, de opacidad… cualquier cosa que se nos ocurra. También se pueden combinar entre sí.

Las transiciones necesitan 4 atributos: propiedad, duración y tipo de transición y demora.

Propiedad:

Se puede aplicar la transición a una única propiedad (p.e. background) o bien a todas las propiedades de un elemento (all).

```
transition-property: nom_propiedad;
```

Duración:

Tenemos que indicar cuantos segundos dura la transición.

```
transition-duration: duración;
```

Demora (opcional):

Si queremos que la transición no empiece inmediatamente, podemos implementar una demora.

```
transition-delay: demora;
```

Tipos de Transiciones:

Siguen el patrón de las Curvas de Bezier[39]

Valores posibles[40]: default, linear, ease-in, ease-out, ease-in-out, cubic-bezier (estableciendo nosotros los valores).

```
transition-timing-function: tipo_transición;
```

Forma abreviada:

Es más útil indicar todas las propiedades de transiciones en una única declaración:

```
transition: nom_propiedad duración tipo_transición retraso;
```

Ejemplo 1: transición de color:

Inicialmente, nuestro bloque tiene un color de fondo. Al realizar un :hover sobre el bloque, éste cambiará de color.

Pero este cambio no se realizará instantáneamente, sino que tardará 2 segundos.

[39] http://www.netzgesta.de/dev/cubic-bezier-timing-function.html
[40] http://www.the-art-of-web.com/css/timing-function/

```
#bloque-transicion-1{
    background:#CCC;
    -webkit-transition: background 2s linear;
}

#bloque-transicion-1:hover{
    background:#333;
}
```

Ejemplo 2: transición de movimiento:

Inicialmente, nuestro bloque está en una determinada posición. Al realizar un :hover sobre el bloque, éste cambiará de posición. El cambio se realizará en 2 segundos.

```
#bloque-transicion-2{
    -webkit-transition: left 2s linear;
}

#bloque-transicion-2:hover{
    left:200px;
}
```

También podemos utilizar generadores[41] automáticos[42].

[41] http://matthewlein.com/ceaser/
[42] http://westciv.com/tools/transforms/index.html

18. KEYFRAMES

Las transformaciones/animaciones básicas que hemos visto, sólo permiten animar un elemento con un determinado tipo de transición.

Podemos añadir más complejidad, juntando varias transiciones en un mismo elemento[43], incluso interactuando varios elementos entre sí[44]. Para ello crearemos nuestra propia animación[45], a la cual invocaremos desde un elemento.

Declaración e invocación:

```
@keyframes MiAnimacion {
    0%   { opacity: 0; }
    100% { opacity: 1; }
}
#bloque {
    animation: MiAnimacion 5s infinite;
}
```

Si queremos que la animación tenga las mismas propiedades al inicio y al final, podemos agrupar los % en una misma declaración::

```
@keyframes MiAnimacion {
    0%, 100% {
       font-size: 10px;
    }
    50% {
       font-size: 12px;
    }
}
```

Propiedades

```
animation-name: MiAnimacion;
animation-duration: 4s;
animation-iteration-count: 10;
animation-direction: alternate;
animation-timing-function: ease-out;
animation-fill-mode: forwards;
animation-delay: 2s;
```

timing-function	ease, ease-out, ease-in, ease-in-out, linear, cubic-bezier(x1, y1, x2, y2) (e.g. cubic-bezier(0.5, 0.2, 0.3, 1.0))
duration & delay	Xs / Xms
duration-count	X
fill-mode	forwards, backwards, both, none
animation-direction	normal, alternate

Podemos también escribir las propiedades resumidas. El orden no importa a excepción de la duración y el retraso. El primer tiempo será siempre el de la duración y el segundo, el del retraso:

```
animation: MiAnimacion 5s 1s 2 alternate backwards
```

[43] http://coding.smashingmagazine.com/2011/05/17/an-introduction-to-css3-keyframe-animations/
[44] http://www.standardista.com/forms/animation.html
[45] https://developer.mozilla.org/en/css/css_animations

Es posible combinar transformaciones con animaciones:

```
@keyframes MiAnimacion {
  from {
    -webkit-transform: rotate(0deg);
  }
  to {
    -webkit-transform: rotate(360deg);
  }
}
```

Uno de los usos principales de las animaciones sencillas va a ser la creación de banners[46] animados[47] [48].

También podemos usar librerías de pequeñas animaciones[49].

Animaciones Múltiples

Podemos crear animaciones múltiples, si separamos las invocaciones por comas:

```
.bloque {
   animation:
      MiAnimacion 2s infinite,
      MiOtraAnimacion 1s;
}
```

[46] http://tympanus.net/Tutorials/AnimatedWebBanners/index.html
[47] http://red-team-design.com/awesome-css3-animated-header/
[48] http://pehaa.com/wp-content/uploads/2011/08/demo/cartedeParis1.html
[49] https://daneden.github.io/animate.css/

19. MEDIA QUERY

Debido a la multitud de dispositivos desde los que accedemos a internet, las posibles dimensiones de la pantalla donde el usuario visualiza una web se han disparado. De ahí que haya nacido el concepto de Responsive Web Design.

Para que la una web se adapte a cada uno de los dispositivos, CSS3 permite comprobar diferentes valores del dispositivo[50]:

- width / height
- device-width / device-height
- orientation (evitar)
- aspect-ratio / device-aspect-ratio
- color / color-index / monochrome
- resolution

Ejemplo 1:

```
<link rel="stylesheet" type="text/css" href="estilos.css"
media="screen and (max-width: 980px)">
```

Ejemplo 2:

```
@media screen and (max-width: 980px) {
    #contenido {width: 60%;}
    #sidebar {width: 30%;}
}
```

Además es necesario indicar explícitamente a los navegadores que estamos utilizando mediaqueries y nuestra página tiene intención de ser responsive.

```
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

Existen páginas para comprobar el soporte/valores de nuestro dispositivo [51] [52] [53].

Y diferentes herramientas para testear nuestras páginas [54] [55] [56].

[50] http://www.w3.org/TR/css3-mediaqueries/
[51] http://jordanm.co.uk/palmreader/
[52] http://mqtest.io/
[53] http://supportdetails.com/
[54] http://mattkersley.com/responsive/
[55] http://quirktools.com/screenfly/
[56] http://www.responsinator.com/

20. SELECTORES AVANZADOS

* E – Elemento

* att – Atributo

* val – Valor

* n – Cualquier número entero.

Sintaxis	Descripción
`E [att^='val']`	Selecciona todos los elementos que inicien con un valor especificado. **Ejemplo:** a[href='http://miweb.com'] Seleccionará todos los enlaces hacia miweb.com
`E[att$='val']`	Selecciona todos los elementos que terminen con un valor especificado. **Ejemplo:** a[href$='.rar'] Seleccionará todos los enlaces de archivos rar.
`E[att*='val']`	Selecciona todos los elementos que contenga, indiferente a su ubicación, un valor especificado. **Ejemplo:** a[href*='tag'] Seleccionará todos los enlaces que contengan la palabra "tag".
`E:root`	Selecciona el elemento raíz de un documento. En HTML, el elemento raíz es siempre <html>
`E:nth-child(n)`	Selecciona el enésimo hijo de su elemento padre. **Ejemplo:** p a:nth-child(2) seleccionará el segundo enlace de un párrafo.
`E:nth-last-child(n)`	Selecciona el enésimo hijo de su elemento padre empezando a constar del último al primero. **Ejemplo:** p a:nth-last-child(2) seleccionará el penúltimo enlace de un párrafo.
`E:nth-of-type(n)`	Selecciona el enésimo elemento de su tipo. **Ejemplo:** p img:nth-of-type(1) seleccionará la primer imagen en un párrafo.
`E:nth-last-of-type(n)`	Selecciona el enésimo elemento de su tipo, empezando a constar del último al primero. **Ejemplo:** p img:nth-last-of-type(1) seleccionará la última imagen en un párrafo.
`E:last-child`	Selecciona el último hijo de un elemento. **Ejemplo:** .post p:last-child seleccionará el último párrafo dentro de un elemento con clase "post".
`E:first-of-type`	Selecciona el primer elemento de su tipo en el elemento padre. **Ejemplo:** .post img:first-of-type seleccionará la primer imagen dentro en un elemento con clase "post".
`E:last-of-type`	Selecciona el último elemento de su tipo en el elemento padre. **Ejemplo:** .post img:last-of-type seleccionará la última imagen dentro en un elemento con clase "post".

`E:only-child`	Selecciona el único elemento hijo de un elemento padre. **Ejemplo:** ul li:only-child seleccionará el elemento una de lista, cuando la lista solo contenga un elemento.
`E:only-of-type`	Selecciona el único elemento de cierto tipo. **Ejemplo:** .post img:only-of-type seleccionará las imágenes presentes en los elementos .post con solo una imagen.
`E:empty`	Selecciona los elementos web que no tienen elementos hijos. **Ejemplo:** ul:empty seleccionará todas las listas no numeradas sin elementos.
`E:target`	Selecciona los elementos que tienen como destino una URL .
`E:enabled`	Selecciona elementos (de formularios) con valor enabled (habilitado). **Ejemplo:** input[type="text"]:enabled seleccionará los campos de tipo texto que están habilitados.
`E:disabled`	Selecciona elementos (de formularios) con valor disabled (deshabilitado). **Ejemplo:** input[type="text"]:disabled seleccionará los campos de tipo texto que están deshabilitados.
`E:checked`	Selecciona elementos (de formularios) con valor checked (seleccionado). **Ejemplo:** input:checked seleccionará los campos que estén seleccionados.
`E::selection`	Selecciona los elementos que han sido seleccionados/resaltados por el usuario. Las propiedades aplicables son color, background, cursor y outline. **Ejemplo:** ::selection ubicara el texto que haya sido seleccionado/resaltado (con el cursor del mouse) por el autor.
`E:not(s)`	Selecciona todo los elementos que no sean un selector indicado (s) dentro de un elemento web. **Ejemplo:** .post:not(img) seleccionará todos los elementos que no sean imágenes y estén dentro del elemento web con clase "post".
`E ~ F`	Selecciona cualquier elemento F que esté precedido por el elemento E

Existen varios simuladores de selectores CSS 3 [57] [58]

[57] https://css-tricks.com/examples/nth-child-tester/
[58] https://css-tricks.com/useful-nth-child-recipies/

RECURSOS Y EJEMPLOS

1. RECURSOS ÚTILES

Generadores automáticos

- Enjoy CSS[59]
- Create CSS[60]
- CSS Generator[61]
- CSS Maker[62]
- CSS Gen[63]

[59] http://enjoycss.com/
[60] http://www.createcss3.com/
[61] http://css3generator.com/
[62] http://www.css3maker.com/
[63] https://css3gen.com/

2. VARIOS EJEMPLOS

Ejemplos

- Recopilatorio[64] / Sistema Solar[65]
- Con Canvas[66]
- Vinilos[67]

Ejemplos (animaciones)

- Gravedad[68]
- Futurama[69]
- Super Mario[70]
- Star Wars intro[71] / Star Wars AT-AT[72]
- Spiderman[73] / The man from Hollywood[74]

Ejemplos (tipografías)

- Back to the future[75]
- Posters parte 1[76] y parte 2[77]

Ejemplos (imágenes)

- Imágenes[78]
- Rueda de color[79]
- Taza de café[80]

Y más ejemplos (recopilación)

- Demos parte 1[81], parte 2[82] y parte 3[83]

[64] https://developer.mozilla.org/en-US/docs/Web/Demos_of_open_web_technologies
[65] http://neography.com/journal/our-solar-system-in-css3/
[66] http://www.effectgames.com/demos/canvascycle/
[67] http://www.zurb.com/playground/sliding-vinyl
[68] http://mrdoob.com/projects/chromeexperiments/google_gravity/
[69] http://www.cssplay.co.uk/menu/css3-animation.html
[70] http://cordobo.com/1662-pure-css-animated-3d-super-mario-icon/
[71] http://www.gesteves.com/experiments/starwars.html
[72] http://anthonycalzadilla.com/css3-ATAT/index.html
[73] http://www.optimum7.com/css3-man/
[74] http://lab.tylergaw.com/themanfromhollywood/
[75] https://code.garron.us/css/BTTF_logo/
[76] http://graphicpush.com/css3-poster-with-no-images
[77] http://neography.com/experiment/type1/
[78] http://www.queness.com/post/4023/18-brilliant-pure-css-drawings
[79] http://crysodenkirk.com/blog/2010/07/color-wheels-with-only-css3-and-primary-colors/
[80] http://gabri.me/htmlcss/2010/css3-gradients-coffee-cup/
[81] http://coding.smashingmagazine.com/2010/07/12/css3-design-contest-results/
[82] http://speckyboy.com/2010/05/21/10-mind-blowing-experimental-css3-techniques-and-demos/
[83] http://webdesignerwall.com/trends/47-amazing-css3-animation-demos