Diseño de Interfaces Web

Unidad 5. Implantación de contenido multimedia

Animaciones, transformaciones y transiciones

Contenido

ANIMACIONES, TRANSFORMACIONES Y TRANSICIONES.

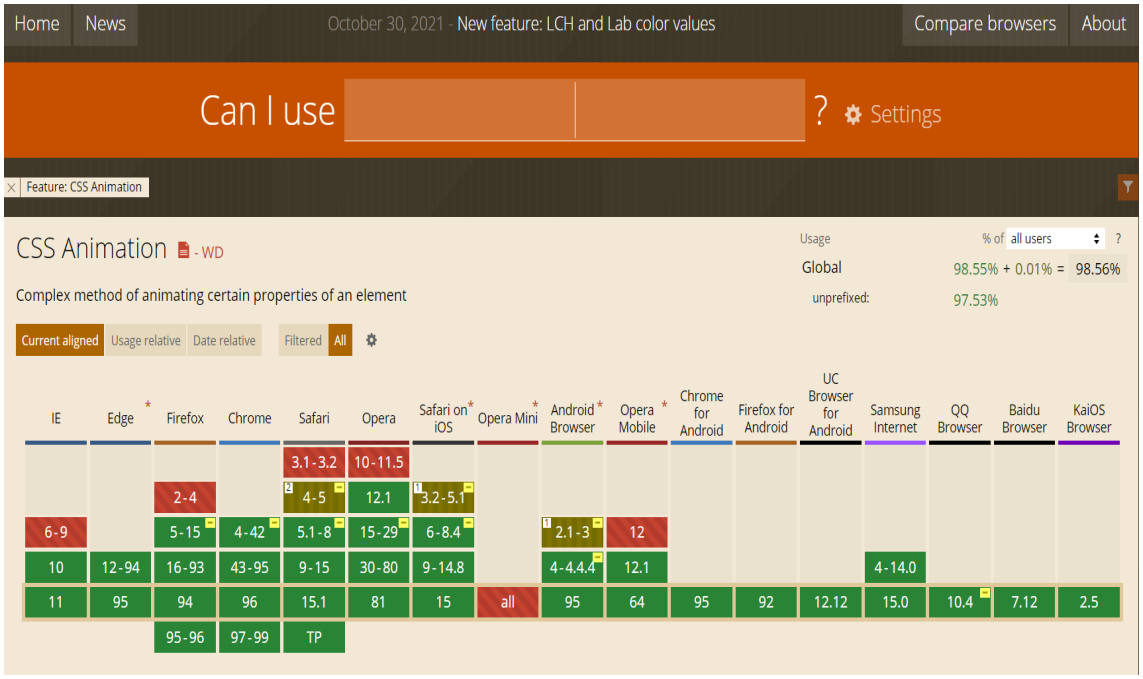
EL MÓDULO ANIMATIONS

1. OBJETIVO DEL MÓDULO

El W3C propone el módulo **Animations**, que está como **Working Draft**: http://www.w3.org/TR/css3-animations/ en octubre de 2018.

Este módulo tiene como objetivo crear animaciones en las páginas web sin que sea preciso recurrir a ningún lenguaje suplementario: ni Flash ni JavaScript.

Según el sitio **Can I use** (http://caniuse.com/#feat=css-animation), la gran mayoría de los navegadores recientes reconocen este módulo y requieren la utilización de prefijos.



2. LA CONSTRUCCIÓN DE LAS ANIMACIONES

Las animaciones se desarrollan en el tiempo. Este último se administra mediante «imágenes claves», las **keyframes** en inglés. Una imagen clave es una etapa en una animación donde el elemento animado opera algún cambio.

La regla @keyframes gestiona las etapas de la animación. Cada animación tendrá una regla @keyframes a la que se habrá dado un nombre.

Usando imágenes claves, deberá determinar las diferentes etapas de la animación e indicar los cambios que se deben aplicar a las distintas propiedades CSS que va a utilizar. Cada etapa podrá identificarse con un valor de porcentaje, sabiendo que el 100 % representa la duración total de la animación.

Puede indicar cuánto tiempo ha de durar la animación con la propiedad **animation-duration**.

La propiedad **animation-timing-function** gestiona la cinética de la animación. Puede usar aceleraciones y ralentizaciones de forma similar a lo que veremos con las transiciones.

A continuación puede indicar cuántas veces debe repetirse esta animación (noción de iteración) usando la propiedad **animation-iteration-count**.

La propiedad **animation-delay** señala si la animación debe comenzar tras una cierta demora, en vez de inmediatamente.

La propiedad **animation-direction** sirve para indicar si la animación debe ejecutarse en sentido inverso (del final al principio).

La propiedad **animation-play-state** señala si la animación se ha reproducido o si se ha interrumpido.

La propiedad **animation-fill-mode** define qué valores aplica la animación fuera del tiempo en que se está ejecutando.

Es posible utilizar la sintaxis corta **animation**, indicando, por este orden:

- animation-name,
- animation-duration,
- animation-timing-function,
- animation-delay,
- animation-iteration-count,
- animation-direction,
- animation-fill-mode
- animation-play-state

3. ANIMAR UNA FORMA

En este ejemplo (**11_01.html**), vamos a desplazar una caja roja de izquierda a derecha y de derecha a izquierda en un bucle sin fin.

El cuadrado rojo es una simple caja :

```
#caja {
   background-color: red;
   position: relative;
   width: 100px;
   height: 100px;
```

```
}
```

La animación se denomina **redBox** (propiedad **animation-name**), dura **5 segundos** (propiedad **animation-duration**) y **se repite sin fin** (propiedad **animation-iteration-count**):

```
#caja1 {
   background-color: red;
   position: relative;
   width: 100px;
   height: 100px;
   -webkit-animation-name: redBox;
   -webkit-animation-duration: 5s;
   -webkit-animation-iteration-count: infinite;
   animation-name: redBox;
   animation-duration: 5s;
   animation-iteration-count: infinite;
}
```

He aquí la sintaxis con la forma corta:

```
#caja {
   background-color: red;
   position: relative;
   width: 100px;
   height: 100px;
   -webkit-animation: redBox 5s infinite;
   animation: redBox 5s infinite;
}
```

En lo que respecta a la animación en sí, el objetivo es que el cuadrado rojo haga un trayecto de ida-vuelta, primero de izquierda a derecha y luego de derecha a izquierda. La vuelta se iniciará cuando la duración de la animación alcance la mitad.

Tenemos, por tanto, tres imágenes claves: a 0 %, a 50 % y a 100 %. En cada etapa indicaremos la posición que debe alcanzarse con la propiedad **left**.

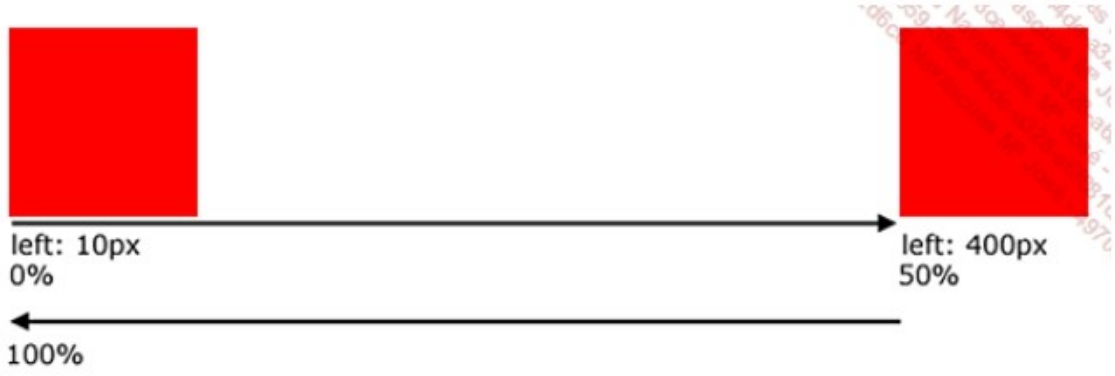
```
@-webkit-keyframes redBox {
   0% {
      left: 10px;
   }
   50% {
      left: 400px;
   }
   100% {
      left: 10px;
   }
}
@keyframes redBox {
   0% {
      left: 10px;
   }
   50% {
      left: 400px;
   }
   100% {
      left: 10px;
   }
}
```

Este es el código completo de la página:

```
<!DOCTYPE HTML>
<html lang="es">
<head>
<title>Cuadrado rojo animado</title>
<meta charset="UTF-8" />
<style>
   #caja {
      background-color: red;
      position: relative;
      width: 100px;
      height: 100px;
      -webkit-animation: redBox 5s infinite;
      animation: redBox 5s infinite;
   }

   @-webkit-keyframes redBox {
      0% {
         left: 10px;
      }
      50% {
         left: 400px;
      }
      100% {
         left: 10px;
      }
   }
   @keyframes redBox {
      0% {
         left: 10px;
      }
      50% {
         left: 400px;
      }
      100% {
         left: 10px;
      }
   }
</style>
</head>
<body>
   <div id="caja"><p> </p></div>
</body>
</html>
```

Veamos el esquema de la animación obtenida:



4. UN CASO PARTICULAR: LOS SPRITES.

En la siguiente página hay desarrollado un ejemplo de animación de un personaje mediante la técnica de sprite.

http://www.emezeta.com/articulos/animar-personajes-con-animaciones-css

EL MÓDULO TRANSFORMS

1. OBJETIVO DEL MÓDULO

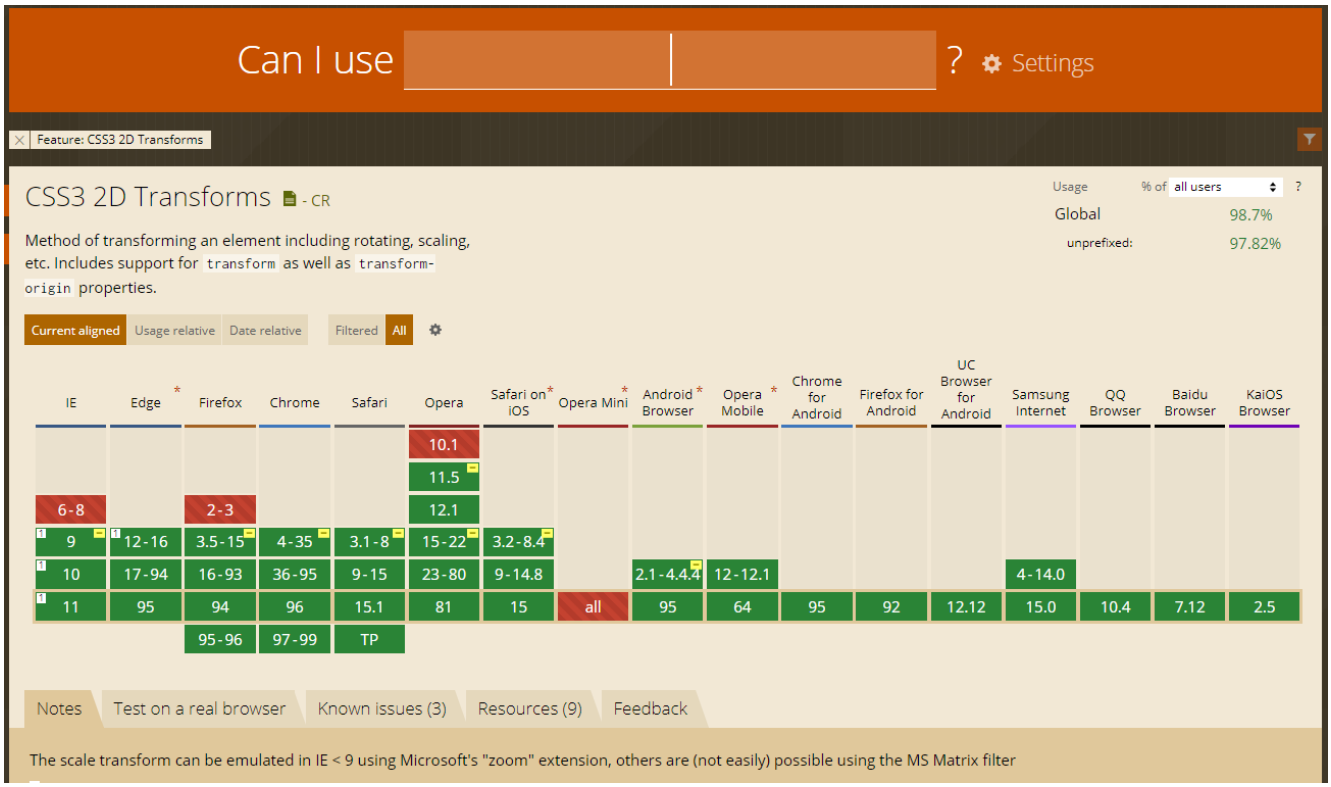
El W3C propone el módulo **Transforms Module Level 1**, que está como **Working Draft**: http://www.w3.org/TR/css3-transforms/, en febrero de 2019.

Las transformaciones permiten modificar, usando propiedades CSS, la visualización de los elementos HTML de una página, en un modelo bi o tridimensional (CSS transforms allows elements styled with CSS to be transformed in two-dimensional or three-dimensional space.). Tras aparecer con la forma habitual, un elemento HTML podrá mostrar transformaciones de tipo rotación, desplazamiento, deformación, escala y perspectiva.

Según el sitio **Can I use** (http://caniuse.com/#feat=transforms2d), la práctica totalidad de los navegadores recientes reconocen la parte 2D de este módulo y únicamente Safari requiere la utilización de prefijos.

2. EJEMPLOS DE TRANSFORMACIÓN

Para llevar a cabo una transformación, es preciso utilizar la propiedad **transform**. Esta propiedad, a continuación, **usará funciones** para aplicar una transformación determinada.



Por defecto, todas las trasformaciones tienen como punto de referencia el centro del elemento. Este punto de referencia es el que se utiliza como punto de partida para calcular las transformaciones.

La propiedad **transform-origin** sirve para cambiar este punto de referencia.

He aquí las transformaciones que se pueden aplicar:

La función **translate** permite efectuar un desplazamiento, una traslación, en una distancia especificada respecto a su posición original y según el punto de referencia. En este ejemplo, la imagen se desplaza 250 píxeles horizontalmente y 50 píxeles verticalmente:

```
.desplazamiento {
   transform: translate(250px,50px);
}
```

La función **scale** permite escalar un elemento a partir de una escala que va de 0 a 1, y en la que 1 es el tamaño de partida. He aquí un ejemplo de un selector que aplica una escala proporcional al 50 % del tamaño original.

```
.escala {
   transform: scale(.5);
}
```

La función **rotate** permite rotar un elemento. La unidad puede expresarse en grados (deg) o en radianes (rad). El siguiente es un ejemplo con una rotación de 23 grados:

```
.rotacion {
   transform: rotate(23deg);
}
```

La función **skew** permite aplicar una deformación a un elemento sobre los dos ejes. La unidad puede expresarse en grados (deg) o en radianes (rad). He aquí un ejemplo con una deformación horizontal de 20 grados y una deformación vertical de -5 grados:

```
.deformacion {
   transform: skew(20deg,-5deg);
}
```

Puede acumular todas las transformaciones que desee; basta con indicar las funciones que quiere utilizar. El siguiente es un ejemplo con todas las transformaciones:

```
.transformacion {
   transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);
}
```

3. UN EJEMPLO DE GALERÍA DE POLAROID

Vamos a llevar a cabo una galería de fotos de tipo «vintage», al estilo Polaroid, con el papel fotográfico expuesto sobre una «mesa» (**11_02.html**). Cuando el ratón pase por encima de una de las fotos, esta se situará en primer plano, su fondo será más blanco y la sombra se marcará más. Este ejemplo está ampliamente inspirado en el que se expone en este sitio: http://line25.com/tutorials/how-to-create-a-pure-css-polaroid-photo-gallery

Comencemos por la estructura de la galería. Creamos esta estructura en HTML5, con los elementos section, figure y figcaption. El elemento section tiene un identificador (galeria) y cada elemento figure posee una clase (pic-1, pic-2...).

```
<section id="galeria">
   <figure class="pic-1">
      <img src="foto1.jpg" alt="Hipopótamo" />
      <figcaption>Hipopótamo</figcaption>
   </figure>
   <figure class="pic-2">
      <img src="foto2.jpg" alt="Tigre" />
      <figcaption>Tigre</figcaption>
   </figure>
   <figure class="pic-3">
      <img src="foto3.jpg" alt="Cebra" />
      <figcaption>Cebra</figcaption>
   </figure>
   <figure class="pic-1">
      <img src="foto4.jpg" alt="Suricata" />
      <figcaption>Suricata</figcaption>
   </figure>
</section>
```

La galería tiene una anchura fijada y está centrada en la página.

```
   #galeria {
      width: 700px;
      margin: 60px auto;
   }
```

Los elementos figure flotan sobre su izquierda (float: left), tienen márgenes diferenciados para espaciarse (margin: 0 10px 10px 0). El relleno inferior es más amplio con el fin de mostrar la leyenda (padding: 10px 25px 10px). El fondo es gris (background: #ccc). Estos elementos tienen bordes (border: 1px solid #fff) una sombra paralela (box-shadow: 0px 2px 15px #333).

```
#galeria figure {
   float: left;
   margin: 0 10px 10px 0;
   padding: 10px 10px 25px 10px;
   background: #ccc;
   border: 1px solid #fff;
   box-shadow: 0px 2px 15px #333;
}
```

El texto de la leyenda está centrado.

```
#galeria figcaption {
   text-align: center;
}
```

Cada foto se halla dentro de un elemento figure que posee una clase dedicada. Para cada clase indicamos un orden de superposición (z-index: 1). Cada foto soporta una transformación con un valor de rotación específico (transform: rotate(-10deg)).

```
#galeria figure.pic-1 {
   z-index: 1;
   -webkit-transform: rotate(-10deg);
   transform: rotate(-10deg);
}
#galeria figure.pic-2 {
   z-index: 2;
   -webkit-transform: rotate(3deg);
   transform: rotate(3deg);
}
#galeria figure.pic-3 {
   z-index: 3;
   -webkit-transform: rotate(5deg);
   transform: rotate(5deg);
}
#galeria figure.pic-4 {
   z-index: 4;
   -webkit-transform: rotate(-10deg);
   transform: rotate(-10deg);
}
```

Ahora vamos a ocuparnos de la transformación. Primero detectamos que el ratón pasa por encima (:hover) de los elementos figure. Modificamos la propiedad z-index para que sea superior a la actual, con objeto de que esa imagen pase por encima de las demás. Aplicamos una sombra más oscura (box-shadow: 3px 5px 15px #000), un color de fondo blanco y una rotación.

```
#galeria figure:hover {
   z-index: 10;
   -webkit-transform: rotate(-3deg);
   transform: rotate(-3deg);
   -webkit-box-shadow: 3px 5px 15px #000;
   box-shadow: 3px 5px 15px #000;
```

```
    background-color: #fff;
}
```

La página presenta este aspecto cuando se carga:

Cuando el ratón pasa por encima de una foto, esta se pone delante de las demás, su fondo se vuelve blanco, la sombra se marca más y se aplica la rotación:

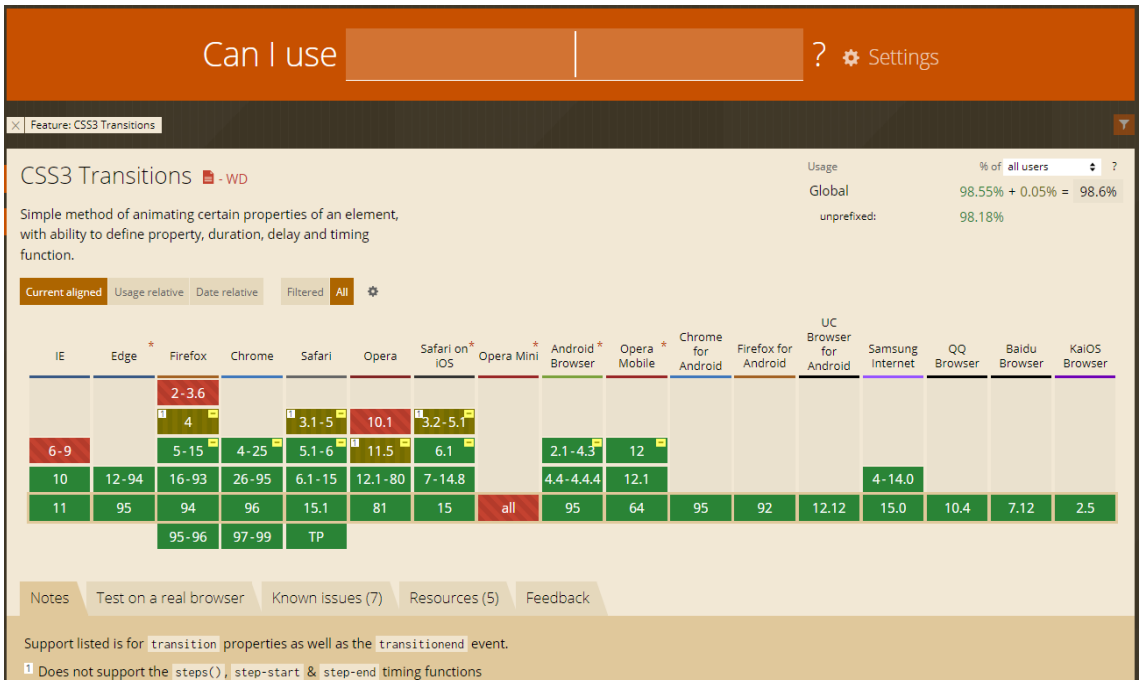
EL MÓDULO TRANSITIONS

1. OBJETIVO DEL MÓDULO

El W3C propone el módulo **Transitions**, que está como **Working Draft**: http://www.w3.org/TR/css3-transitions/, en octubre de 2018.

Las transformaciones permiten pasar de un valor de CSS a otro con una transición especificada cuando se detecta un evento en un elemento. Cuando la transición ha finalizado, el elemento retoma sus parámetros CSS iniciales.

El sitio **Can I use** (http://caniuse.com/#feat=css-transitions) muestra que el reconocimiento de este módulo por parte de los navegadores es excelente, sin que sea necesario utilizar los prefijos.



2. PONER EN MARCHA LAS TRANSICIONES

Para desencadenar una transición, es preciso detectar un evento, por ejemplo con una pseudoclase: :hover, :active o :focus.

Debe indicar qué propiedades CSS desea utilizar con la propiedad **transition-property**.

La lista de las propiedades que pueden usarse con las transiciones es bastante amplia. Consúltela en el sitio del W3C: http://www.w3.org/TR/css3-transitions/#animatable-css

Observe que la propiedad transition-property acepta también dos palabras claves preestablecidas:

- **transition-property: none** indica que no hay ninguna propiedad implicada en las transiciones,
- **transition-property: all** indica que todas las propiedades son susceptibles de usarse en las transiciones.

La propiedad **transition-duration** permite determinar la duración de la transición.

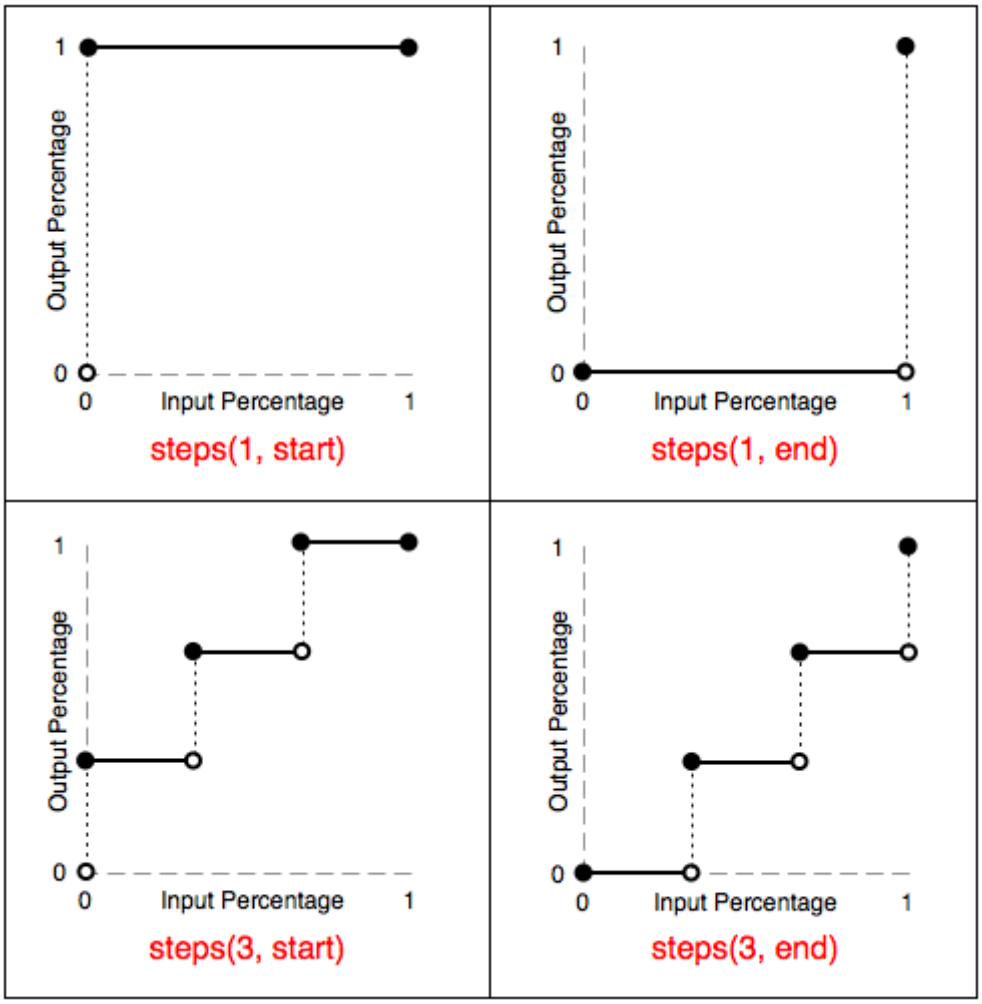
La propiedad **transition-timing-function** permite determinar la cinética de la transición. Puede ser cualquiera de las siguientes palabras clave: **ease, linear, ease-in, ease-out o ease-in-out**. Estas palabras clave determinan cómo se realizará el proceso de transición basado en una curva Bézier. Cada una de ellas representa diferentes tipos de curva Bézier con las siguientes características:

- **ease**: especifica un efecto de transición con un arranque lento, luego rápido, luego finaliza lentamente (es el valor predeterminado).
- **linear:** especifica un efecto de transición con la misma velocidad de principio a fin.
- **ease-in:** especifica un efecto de transición con un inicio lento.
- **Ease-out**: especifica un efecto de transición con un final lento.

- **Ease-in-out**: especifica un efecto de transición con un inicio y fin lentos.
- **Cubic-bezier (n, n, n, n):** permite definir sus propios valores en una función cúbico-bezier.

Valor	Inicio	Transcurso	Final	Equivalente en cubic-beizer
ease	Lento	Rápido	Lento	(0.25, 0.1, 0.25, 1)
linear	Normal	Normal	Normal	(0, 0, 1, 1)
ease-in	Lento	Normal	Normal	(0.42, 0, 1, 1)
ease-out	Normal	Normal	Lento	(0, 0, 0.58, 1)
ease-in-out	Lento	Normal	Lento	(0.42, 0, 0.58, 1)
cubic-bezier(*A*, *B*, *C*, *D*)	-	-	-	Transición personalizada

También es posible usar la función **steps(n)** que permite dividir el proceso de transición en intervalos de igual duración.



La propiedad **transition-delay** permite determinar cuándo comienza la transición (cuánto tiempo debe transcurrir tras la detección del evento).

Puede utilizar la sintaxis corta **transition** especificando a continuación las funciones, separadas por un espacio, en este orden:

- transition-property,
- transition-duration,
- transition-timing-function,
- transition-delay.

He aquí un ejemplo de transición corta:

```
#desplazamiento {
   position: absolute;
   left: 20px;
   top: 20px;
   transition: left 2s ease-in 2s;
}
```

3. UN EJEMPLO DE MENÚ INTERACTIVO

Vamos a crear un menú muy sencillo, con dos transiciones que van a intervenir cuando el ratón pase por encima de los enlaces:

- el color de fondo pasará progresivamente del amarillo al azul,
- el color del texto va a pasar progresivamente del negro al blanco.

Para empezar, creamos un menú tradicional con una lista <ul>:

```
<ul id="navegacion">
   <li><a href="#">Inicio</a></li>
   <li><a href="#">Referencias</a></li>
   <li><a href="#">Contacto</a></li>
</ul>
```

El formato del menú también es tradicional, con un fondo amarillo (background-color: lightyellow;) y un texto negro (color: black;):

```
#navegacion li {
   list-style: none;
}
#navegacion a {
   display: block;
   float: left;
   width: 100px;
   padding: 3px;
   margin-right: 20px;
   border: 1px solid #333;
   background-color: lightyellow;
   text-align: center;
   color: black;
   text-decoration: none;
}
```

He aquí el aspecto del menú cuando se carga la página:

Ahora crearemos las dos transiciones sobre las propiedades background-color y color. Estas dos transiciones duran 1 segundo y son lineales.

```
#navegacion a {
   display: block;
   float: left;
   width: 100px;
   padding: 3px;
   margin-right: 20px;
   border: 1px solid #333;
   background-color: lightyellow;
   text-align: center;
   color: black;
   text-decoration: none;
```

```
   /* Las transiciones */
   transition: background-color 1s linear, color 1s linear;
}
```

Ahora introducimos el evento: el ratón que pasa por encima (:hover) de los vínculos a:

```
#navegacion a:hover {
   background-color: darkblue;
   color: white;
}
```

El color de fondo de los vínculos pasará a ser azul oscuro: background-color: darkblue.

El color del texto de los vínculos a pasará a blanco: color: white.

He aquí el código completo de este ejemplo (**11_03.html**):

```
<html lang="es">
<head>
<title>Las transiciones</title>
<meta charset="UTF-8" />
<style>
body {
   padding: 20px;
   margin: 0;
}
#navegacion li {
   list-style: none;
}
#navegacion a {
   display: block;
   float: left;
   width: 100px;
   padding: 3px;
   margin-right: 20px;
   border: 1px solid #333;
   background-color: lightyellow;
   text-align: center;
   color: black;
   text-decoration: none;
   /* Las transiciones */
   transition: background-color 1s linear, color 1s linear;
}
#navegacion a:hover {
   background-color: darkblue;
   color: white;
}
</style>
</head>
<body>
<ul id="navegacion">
   <li><a href="#">Inicio</a></li>
   <li><a href="#">Referencias</a></li>
   <li><a href="#">Contacto</a></li>
</ul>
</body>
</html>
```

Cuando el puntero pasa por encima de un vínculo, el fondo de su caja se vuelve azul oscuro, mientras que el texto pasa de negro a blanco: