2. Animaciones CSS3

Tradicionalmente, cuando queríamos incluir cualquier tipo de animación en nuestras páginas web, teníamos que recurrir a tecnologías como Flash o JavaScript. Si bien la potencia de ambos es descomunal, las nuevas funciones de animación de CSS3 nos facilitan mucho la tarea de crear animaciones sin tener que depender de tecnologías de terceros y sin necesidad de programar.

Ya vimos en el tema 3 como podíamos crear pequeñas animaciones, utilizando para ello las transiciones de CSS3. En este tema, iremos un paso más allá, y encadenaremos múltiples transiciones juntas en la misma propiedad para que sean realizadas una tras otra.

Las animaciones constan de dos componentes: un estilo que describe la animación y 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:

1. Es muy fácil crear animaciones sencillas, puedes hacerlo incluso sin tener conocimientos de javascript.
2. La animación se muestra correctamente, incluso en equipos poco potentes. Animaciones simples realizadas en javascript pueden verse mal (a menos que estén muy bien programadas). El motor de renderizado puede usar técnicas de optimización como el "frame-skipping" u otras para conseguir que la animación se vea tan suave como sea posible.
3. Al ser el navegador quien controla 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.

Fotogramas claves de las animaciones CSS3

Para aquellos que hayáis trabajado alguna vez con Flash, el concepto de fotograma clave no necesita explicación. Los fotogramas clave de las animaciones CSS3 son muy similares a los que encontramos en Flash.

Un fotograma clave no es más que un punto destacado en el tiempo de nuestra animación. Cada fotograma describe cómo se muestra cada elemento animado en

un momento dado durante la secuencia de la animación. Cualquier animación consta al menos de dos fotogramas claves: el punto inicial y el punto final. Imaginad que nuestra animación es como una carretera:

El primer semáforo actuaría como fotograma clave inicial y el segundo como el final. Entre uno y otro se produciría nuestra animación, que no es más que el desplazamiento del coche hacia la derecha.

@keyframes

En CSS3 creamos animaciones completas mediante `@keyframes`, que son un conjunto de fotogramas clave. Su sintaxis es la siguiente:

```
@keyframes nombreAnimacion{
     puntoDelKeyframe{
          atributosIniciales;
     }
     puntoDelKeyframe{
          nuevosAtributos;
     }

     .......

     puntoDelKeyframe{
         últimosAtributos;
     }
}
```

Visto así, no queda demasiado claro, pero veamos que tendríamos que hacer para desplazar nuestro coche a la derecha:

```
@keyframes animacionCoche
{
     /*Indicamos que salimos de la posición 0*/
     from
     {
          left:0px;
     }
     /*Indicamos que al final la posición debe ser 350*/
     to
     {
          left:350px;
     }
}
```

Los fotogramas usan porcentajes para indicar en qué momento de la secuencia de la animación tienen lugar: `0%` es el principio, `100%` es el estado final de la animación. Obligatoriamente, debemos especificar estos dos fotogramas para que el

navegador sepa dónde debe comenzar y finalizar; debido a su importancia, estos dos fotogramas tienen alias especiales: `from` y `to` (ver el ejemplo anterior).

Podemos crear animaciones más complejas estableciendo fotogramas claves intermedios mediante porcentajes:

```
@keyframes animacionCoche
{
     from
     {
          left:0px;
     }
 /*Hasta el 65% de la reproducción solo queremos que se desplace 10 píxeles*/
     65%
     {
          left:10px;
     }
     to
     {
          left:350px;
     }
}
```

Compatibilidad entre navegadores

Actualmente, para que las animaciones funcionen correctamente en todos los navegadores, debemos utilizar los prefijos propietarios de cada navegador.

```
@keyframes nombreAnimacion { … }
@-webkit-keyframes nombreAnimacion { … }
@-moz-keyframes nombreAnimacion { … }
@-o-keyframes nombreAnimacion { … }
```

Además, no podremos separar las distintas declaraciones por comas, todas se deben hacer aparte. El siguiente código no funcionaría:

```
@keyframes nombreAnimacion, @-webkit-keyframes nombreAnimacion,
@-moz-keyframes nombreAnimacion, @-o-keyframes nombreAnimacion,
@-ms-keyframes nombreAnimacion { … }
```

Asignar animaciones CSS3 a un elemento

Para asignar una secuencia de animación CSS3 a un elemento utilizaremos la propiedad `animation` y sus sub-propiedades. Con ellas podemos no sólo configurar el ritmo y la duración de la animación, sino otros detalles sobre la secuencia de la animación.

Con estas propiedades no configuramos la apariencia actual de la animación, para eso disponemos de `@keyframes` (visto anteriormente). Lo que haremos mediante estas propiedades es asociar una secuencia de animación, creada anteriormente

con `@keyframes`, a un elemento y configurar cómo queremos que se reproduzca la animación.

Las subpropiedades de `animation` son:

- `animation-delay:` tiempo de retardo en segundos entre el momento en que el elemento se carga y el comienzo de la secuencia de la animación.
- `animation-direction`: indica la dirección de la animación. Los posibles valores son:
  - `normal`: La animación se reproduce desde el inicio hasta el final.
  - `reverse`: La animación se reproduce desde el final hasta el inicio.
  - `alternate`: La animación se reproduce normalmente en las iteraciones impares y hacia atrás en las pares.
  - `alternate-reverse`: La animación se reproduce hacia atrás en las iteraciones impares y hacia delante en las pares.
- `animation-duration`: indica la cantidad de tiempo que la animación consume en completar su ciclo (duración).
- `animation-iteration-count`: el número de veces que se repite. Podemos indicar `infinite` para repetir la animación indefinidamente.
- `animation-name`: Especifica el nombre de la regla `@keyframes` que describe los fotogramas de la animación.
- `animation-play-state`: permite pausar y reanudar la secuencia de la animación. Se podría usar desde javascript para pausar la animación.
- `animation-timing-function`: indica el ritmo de la animación, es decir, como se muestran los fotogramas de la animación, estableciendo curvas de aceleración. Por defecto, responde al valor `ease`, pero puede responder a diferentes valores: `ease`, `linear`, `ease-in`, `ease-out`, `ease-in-out`.
- `animation-fill-mode`: especifica qué valores tendrán las propiedades después de finalizar la animación. Los posibles valores son:
  - `none`: Es el valor por defecto. El elemento no tendrá ningún estilo aplicado ni antes ni después de la animación.
  - `forwards`: El elemento se quedará con los estilos aplicados al final de la animación.
  - `backwards`: El elemento se quedará con los estilos aplicados al principio de la animación.

- `both`: El elemento se quedará con los estilos aplicados al principio de la animación y al final de la misma.

Veamos cómo asignaríamos la animación del ejemplo anterior a un elemento de nuestra página que tenga el `id coche`:

```
#coche
{
    animation-duration: 3s;
    animation-name: animacionCoche;
    animation-iteration-count: 1;
    position:relative;
}
```

El estilo del elemento `#coche` indica, a través de la propiedad `animation-duration`, que la animación debe durar 3 segundos desde el inicio al fin y que el nombre de los `@keyframes` que definen los fotogramas de la secuencia de la animación es `animacionCoche`. Además, la animación sólo se reproducirá una vez.

Evidentemente, si quisiéramos que la animación funcione en todos los navegadores, tendremos que utilizar los prefijos correspondientes. Veamos una animación completa, utilizando los prefijos correspondientes:

```
@keyframes animacionCoche
{
    from { transform:rotate(-18deg); }
    65% { transform:rotate(18deg); }
    to { transform:rotate(0deg); }
}

@-webkit-keyframes animacionCoche
{
    from { -webkit-transform:rotate(-18deg); }
    65% { -webkit-transform:rotate(18deg); }
    to { -webkit-transform:rotate(0deg); }
}

@-moz-keyframes animacionCoche
{
    from { -moz-transform:rotate(-18deg); }
    65% { -moz-transform:rotate(18deg); }
    to { -moz-transform:rotate(0deg); }
}

@-o-keyframes animacionCoche
{
    from { -o-transform:rotate(-18deg); }
    65% { -o-transform:rotate(18deg); }
    to { -o-transform:rotate(0deg); }
}

#coche
{
    animation-duration: 3s;
    animation-name: animacionCoche;
    animation-iteration-count: 1;
```

```
    -webkit-animation-duration: 3s;
    -webkit-animation-name: animacionCoche;
    -webkit-animation-iteration-count: 1;

    -moz-animation-duration: 3s;
    -moz-animation-name: animacionCoche;
    -moz-animation-iteration-count: 1;

    -o-animation-duration: 3s;
    -o-animation-name: animacionCoche;
    -o-animation-iteration-count: 1;

    position:relative;
}
```

En este caso, hemos realizado una animación de rotación para que quede claro que los prefijos hay que utilizarlos tanto al crear la animación como en las propiedades que lo requieran.

Como se puede observar, es bastante tedioso crear una animación que sea compatible con todos los navegadores, pero es el precio que tenemos que pagar en aras de la compatibilidad.

En cualquier caso, pese a este inconveniente, sigue siendo mucho más sencillo crear una animación utilizando CSS3 que utilizando javascript.

Controlar la reproducción de las animaciones

Es muy habitual que nos interese reproducir una animación, no cuando se cargue la página, sino como respuesta a un determinado evento. Por ejemplo, nos puede interesar que una animación se inicie cuando colocamos el ratón sobre un elemento. Este caso será el más sencillo de implementar, puesto que disponemos de la pseudo-clase `:hover`. En el ejemplo anterior, si quisiéramos que el coche se moviera sólo cuando ponemos el ratón sobre él, haríamos lo siguiente:

```
#coche:hover
{
    animation-duration: 3s;
    animation-name: animacionCoche;
    animation-iteration-count: 1;

    -webkit-animation-duration: 3s;
    -webkit-animation-name: animacionCoche;
    -webkit-animation-iteration-count: 1;

    -moz-animation-duration: 3s;
    -moz-animation-name: animacionCoche;
    -moz-animation-iteration-count: 1;

    -o-animation-duration: 3s;
    -o-animation-name: animacionCoche;
```

```
    -o-animation-iteration-count: 1;

    position:relative;
}
```

Como vemos, este caso es obvio y no supone ninguna complicación, pero ¿qué sucedería si quisiéramos que una animación se iniciara cuando hacemos click con el ratón sobre un elemento? En este caso, no tendríamos más remedio que recurrir a javascript.

Supongamos que queremos que la animación del coche se inicie cuando pulsamos sobre un botón de la página:

```
<button id="iniciaAnimacion">Inicia animación</button>
<div id="coche"><p>coche</p></div>
```

Vamos a utilizar los siguientes estilos:

```
@keyframes animacionCoche
{
    from { left: 0px; }
    65% { left: 10px; }
    to { left: 350px; }
}

@-webkit-keyframes animacionCoche
{
    from { left: 0px; }
    65% { left: 10px; }
    to { left: 350px; }
}

@-moz-keyframes animacionCoche
{
    from { left: 0px; }
    65% { left: 10px; }
    to { left: 350px; }
}

@-o-keyframes animacionCoche
{
    from { left: 0px; }
    65% { left: 10px; }
    to { left: 350px; }
}

.aCoche
{
    animation-duration: 3s;
    animation-name: animacionCoche;
    animation-iteration-count: 1;

    -webkit-animation-duration: 3s;
    -webkit-animation-name: animacionCoche;
    -webkit-animation-iteration-count: 1;

    -moz-animation-duration: 3s;
    -moz-animation-name: animacionCoche;
```

```
    -moz-animation-iteration-count: 1;

    -o-animation-duration: 3s;
    -o-animation-name: animacionCoche;
    -o-animation-iteration-count: 1;

    position:relative;
}
```

Es importante observar que, en este caso, hemos utilizado un estilo de clase (`.aCoche`) para asignarle la animación al elemento, por lo tanto, la animación se reproducirá cuando le asociemos la clase al elemento que nos interese. A continuación, vamos a ver el código javascript que necesitaremos para asociar la clase `aCoche` con el elemento `coche`:

```
function iniciar()
{
      var iniciaAnimacion=document.getElementById("iniciaAnimacion");
      iniciaAnimacion.addEventListener("click", accIniciaAnimacion, false);
}

function accIniciaAnimacion()
{
      var coche=document.getElementById("coche");
      coche.className = "aCoche";
}

window.addEventListener("load", iniciar, false);
```

En el momento que pulsamos sobre el botón, se le asignará la clase `aCoche` al elemento `coche`, y será en este instante cuando se inicie la animación.

Encadenar animaciones

Hasta ahora hemos visto que las animaciones nos servirán para aplicar varias transiciones consecutivas (una tras otra) sobre un mismo elemento. Muy bien, pero ¿y si quisiéramos aplicar varias transiciones consecutivas, pero no sobre un mismo elemento, sino sobre elementos diferentes? La respuesta es obvia, tendríamos que crear una animación diferente para cada elemento que quisiéramos animar. Perfecto, pero también hemos visto que las animaciones se iniciarán en el momento que se asocie el estilo que contiene la animación al elemento que queremos animar, entonces, ¿cómo haríamos para encadenar animaciones?, es decir, ¿cómo podemos hacer para que se inicie una animación en el momento que finalice la anterior? para hacer esto se han incorporado nuevos eventos para el control de las animaciones. En concreto, disponemos de tres eventos que nos darán información sobre el estado en el que se encuentra la animación. Estos eventos son los siguientes:

- `animationstart`: este evento será lanzado cuando se inicie la animación por primera vez. Si la animación está configurada para hacer varias iteraciones, sólo se lanzará el evento cuando se inicie la primera de ellas.
- `animationiteration`: este evento será lanzado al inicio de cada iteración de la animación, excepto en la primera.
- `animationend`: este evento será lanzado al final la animación.

Con esto, ya podemos encadenar nuestras animaciones, por ejemplo, veamos cómo haríamos para que nuestro coche se desplace a la derecha hasta chocar con otro coche y, en ese momento, el otro coche inicie otro desplazamiento a la derecha.

```
<button id="iniciaAnimacion">Inicia animación</button>
<div id="coche1"><p>coche1</p></div>
<div id="coche2"><p>coche2</p></div>
```

Ahora tenemos dos coches en lugar de uno. Los estilos iniciales serán los siguientes:

```
#coche1
{
    position:absolute;
    top:200px;
    left: 0px;
    width: 50px;
    display:inline;
}

#coche2
{
    position:absolute;
    top:200px;
    left: 400px;
    width: 50px;
    display:inline;
}
```

A continuación, veremos los estilos que crearemos para las animaciones:

```
@keyframes animacionCoche1
{
    from { left: 0px; }
    65% { left: 10px; }
    to { left: 350px; }
}

@-webkit-keyframes animacionCoche1
{
    from { left: 0px; }
    65% { left: 10px; }
    to { left: 350px; }
}

@-moz-keyframes animacionCoche1
{
```

```
    from { left: 0px; }
    65% { left: 10px; }
    to { left: 350px; }
}

@-o-keyframes animacionCoche1
{
    from { left: 0px; }
    65% { left: 10px; }
    to { left: 350px; }
}

.aCoche1
{
    animation-duration: 3s;
    animation-name: animacionCoche1;
    animation-iteration-count: 1;
    animation-fill-mode: forwards;

    -webkit-animation-duration: 3s;
    -webkit-animation-name: animacionCoche1;
    -webkit-animation-iteration-count: 1;
    -webkit-animation-fill-mode: forwards;

    -moz-animation-duration: 3s;
    -moz-animation-name: animacionCoche1;
    -moz-animation-iteration-count: 1;
    -moz-animation-fill-mode: forwards;

    -o-animation-duration: 3s;
    -o-animation-name: animacionCoche1;
    -o-animation-iteration-count: 1;
    -o-animation-fill-mode: forwards;
}

@keyframes animacionCoche2
{
    from { left: 400px; }
    65% { left: 550px; }
    to { left: 600px; }
}

@-webkit-keyframes animacionCoche2
{
    from { left: 400px; }
    65% { left: 550px; }
    to { left: 600px; }
}

@-moz-keyframes animacionCoche2
{
    from { left: 400px; }
    65% { left: 550px; }
    to { left: 600px; }
}

@-o-keyframes animacionCoche2
{
    from { left: 400px; }
```

```
    65% { left: 550px; }
    to { left: 600px; }
}

.aCoche2
{
    animation-duration: 3s;
    animation-name: animacionCoche2;
    animation-iteration-count: 1;
    animation-fill-mode: forwards;

    -webkit-animation-duration: 3s;
    -webkit-animation-name: animacionCoche2;
    -webkit-animation-iteration-count: 1;
    -webkit-animation-fill-mode: forwards;

    -moz-animation-duration: 3s;
    -moz-animation-name: animacionCoche2;
    -moz-animation-iteration-count: 1;
    -moz-animation-fill-mode: forwards;

    -o-animation-duration: 3s;
    -o-animation-name: animacionCoche2;
    -o-animation-iteration-count: 1;
    -o-animation-fill-mode: forwards;
}
```

Finalmente, veremos el código javascript necesario para encadenar las animaciones:

```
function iniciar()
{
      var iniciaAnimacion=document.getElementById("iniciaAnimacion");
      iniciaAnimacion.addEventListener("click", accIniciaAnimacion, false);
}

function accIniciaAnimacion()
{
      var coche1=document.getElementById("coche1");

      coche1.className = "aCoche1";
      coche1.addEventListener("animationend", animacionCoche1Fin, false);
}

function animacionCoche1Fin()
{
      var coche2=document.getElementById("coche2");
      coche2.className = "aCoche2";
}

window.addEventListener("load", iniciar, false);
```

Como vemos en el ejemplo, esperamos a que termine la primera animación para iniciar la siguiente.

Compatibilidad entre navegadores

Como siempre, el problema que nos encontramos es la compatibilidad entre navegadores. Aunque estos eventos ya forman parte del estándar W3C, todavía

tenemos que utilizar prefijos (al estilo de CSS3) para asegurarnos del correcto funcionamiento en todos los navegadores modernos. En la siguiente tabla podemos ver los diferentes prefijos utilizados para cada uno de los navegadores.

W3C standard	Firefox	webkit	Opera	IE10
animationstart	mozAnimationStart	webkitAnimationStart	oanimationstart	MSAnimationStart
animationiteration	mozAnimationIteration	webkitAnimationIteration	oanimationiteration	MSAnimationIteration
animationend	mozAnimationEnd	webkitAnimationEnd	oanimationend	MSAnimationEnd

Para que nuestro código funcione correctamente en todos los navegadores, tendremos que introducir un manejador de evento para cada uno de los diferentes eventos existentes:

```
coche1.addEventListener("animationend", animacionCoche1Fin, false);
coche1.addEventListener("webkitAnimationEnd", animacionCoche1Fin, false);
coche1.addEventListener("mozAnimationEnd", animacionCoche1Fin, false);
coche1.addEventListener("oanimationend", animacionCoche1Fin, false);
coche1.addEventListener("MSAnimationEnd", animacionCoche1Fin, false);
```

De esta forma, el evento que reconozca el navegador será manejado y el resto serán ignorados. Como vemos, es bastante tedioso tener que añadir todos estos eventos, por lo que, utilizaremos una función que lo haga por nosotros:

```
function PrefixedEvent(element, type, callback)
{
	for (var p = 0; p < pfx.length; p++)
	{
		if (pfx[p] == "" || pfx[p] == "o")
			type = type.toLowerCase();
		element.addEventListener(pfx[p]+type, callback, false);
	}
}
```

Esta función generara los diferentes manejadores de evento para todas las posibilidades que existen. Cada vez que tengamos que crear un manejador de evento llamaremos a esta función. Por ejemplo, la llamada anterior quedaría así:

```
PrefixedEvent(coche1, "AnimationEnd", animacionCoche1Fin);
```

Siempre pasaremos el nombre del evento con las iniciales de las palabras en mayúsculas, la función ya se encargará de convertirlas a minúsculas cuando sea necesario.

Utilizando esta función, el ejemplo anterior quedaría de la siguiente forma:

```
function iniciar()
{
     var iniciaAnimacion=document.getElementById("iniciaAnimacion");

     iniciaAnimacion.addEventListener("click", accionIniciaAnimacion, false);
```

```
}

function accionIniciaAnimacion()
{
	var coche1=document.getElementById("coche1");

	coche1.className = "aCoche1";
	PrefixedEvent(coche1, "AnimationEnd", animacionCoche1Fin);
}

function animacionCoche1Fin()
{
	var coche2=document.getElementById("coche2");

	coche2.className = "aCoche2";
}

window.addEventListener("load", iniciar, false);

var pfx = ["webkit", "moz", "MS", "o", ""];

function PrefixedEvent(element, type, callback)
{
	for (var p = 0; p < pfx.length; p++)
	{
		if (pfx[p] == "" || pfx[p] == "o")
			type = type.toLowerCase();
		element.addEventListener(pfx[p]+type, callback, false);
	}
}
```

Podéis encontrar este ejemplo completamente funcional en la carpeta `ejemplo-animacion-coche` del fichero de recursos.

Con esto, damos por finalizados los contenidos del tema. A continuación, plantearemos una serie de ejercicios para practicar todo lo estudiado.