Animación CSS3

Cómo crear animaciones

Una animación es cualquier elemento o grupo de elementos que se mueve o cambia de aspecto progresivamente.

Hasta ahora las animaciones en la web debíamos hacerlas con javascript o con algún programa externo como "flash" o "java".

El proceso para crear una animación puede ser más o menos complicado según el programa que usemos y el tipo de animación. En una animación un elemento cambia de estado, bien sea cambiando de sitio, de tamaño, de color, de forma, etc. Este cambio se realiza de manera gradual, de forma que vemos como el elemento va cambiando, la velocidad de cambio debe ser tal que el ojo la perciba como un movimiento continuo (superior a 10 o 12 cambios por segundo).

Las primeras animaciones se hacían indicando uno a uno cada cambio de estado o posición del elemento, e incluso teniendo que redefinir el elemento en cada punto de ese cambio. Cada uno de los cambios que tiene el elemento desde su estado inicial al final es un fotograma. En lenguajes como javascript debemos definir fotograma a fotograma toda la transición del elemento, aunque para ello nos ayudamos de elementos como los bucles que repiten automáticamente el proceso.

Las animaciones con CSS3 son bastante más sencillas, ya que utilizan un método usado en programas como "flash", el de los fotogramas clave.

Fotograma clave

Los fotogramas clave son los que marcan el estado del elemento animado al principio, al final, o cuando éste tenga que variar de forma no previsible.

En un ejemplo sencillo, si queremos que un elemento se mueva de izquierda a derecha, posicionamos en el primer fotograma el elemento en un punto a la izquierda, y en el segundo fotograma lo posicionamos a la derecha. le indicamos luego el tiempo que tiene que tardar de ir de uno a otro; si queremos que después el elemento siga moviéndose hacia abajo, lo posicionaremos en un tercer fotograma clave, situado en un punto debajo del segundo. El elemento recorrerá en línea recta del primero al segundo fotograma y seguidamente del segundo al tercero. Todos los estados intermedios del elemento entre los fotogramas 1, 2 y 3 se calculan automáticamente. Ahora veamos cómo se hace todo esto con CSS3.

La regla @keyframes

En CSS los fotogramas clave se crean mediante la regla `@keyframes` la cual tiene la siguiente estructura:

```
@keyframes <nombre> {
      0%{<propiedad: valor>+}
      100%{<propiedad: valor>+}
      }
```

En el selector escribimos la palabra clave `@keyframes` seguido de un nombre que nosotros le daremos a la animación. Este nombre es necesario para poder dar después otras propiedades de animación, como puede ser el tiempo.

Dentro de la declaración (espacio entre llaves) pondremos los diversos fotogramas clave. los cuales son unas sub-reglas CSS compuestas de:

- **Selector:** Los selectores indican el punto en el que se coloca el fotograma clave. Se indican en tantos por ciento. El primero debe ser siempre el `0%` que indica el principio de la animación, y el último el `100%` e indica el final. Entre medio se pueden colocar más fotogramas clave, indicando siempre en tanto por ciento el punto en el que estará el fotograma. Cuando hay sólo dos fotogramas pueden usarse también las palabras clave `from` para el primero y `to` para el último.
- **Declaración:** Dentro de las llaves de los fotogramas pondremos las propiedades CSS del elemento en el momento que se visualice el fotograma. Por ejemplo, en el primero marcamos el color, posición y tamaño inicial; en el último marcamos color, posición y tamañó final. Las propiedades, siempre que puedan, (propiedades que contienen números - medida- color - etc) cambiarán gradualmente del primero al último durante el tiempo que dure la animación.

NAVEGADORES: Los principales navegadores admiten en sus últimas versiones La regla `@keyframes`, Sin embargo hasta hace poco tiempo Firefox utilizaba la forma experimental, con el prefijo -moz- (`@-moz-keyframes`). Ultimamente Firefox ha adoptado la forma estándar. Safari y Chrome siguen utilizando la forma experimental con el prefijo -webkit- (`@-webkit-keyframes`). Opera también ha incorporado recientemente la forma estándar. Internet Explorer incorpora también la forma estándar a partir de la versión 10.

Propiedades de animación.

Vamos a ver aquí las propiedades necesarias para crear una animación simple. En primer lugar debemos indicar qué elemento de la página es el que tiene la animación. Esto se hace mediante la propiedad:

```
#anim { animation-name: <nombre> }
```

El "nombre" que pongamos como valor debe ser el mismo que lleva la regla `@keyframe`; de esa manera quedará asociada al elemento que marca el selector.

En segundo lugar debemos marcar el tiempo que dura la animación. Este lo indicamos mediante la propiedad:

```
animation-duration: <tiempo>;
```

El "tiempo" lo expresaremos en segundos (ej.: `10s`) o en milisegundos (ej.: `4000ms`) indicandolo en número seguido de la/s letra/s "s" para segundos o "ms" para milisegundos.

Aunque con esto ya podemos hacer una animación muy sencilla, la animación que vamos a hacer como ejemplo tendrá otras dos propiedades, la primera de ellas hará que la animación se repita indefinidamente:

```
animation-iteration-count: infinite;
```

El valor "infinite" hace que la animación se repita indefinidamente. Además de este valor, a esta propiedad se le puede dar como valor un número entero positivo, que será el número de veces que se repita la animación.

Por último queremos que la animación, una vez llegue al final, recorra el camino inverso, viéndose otra vez todo al revés. Para ello utilizaremos la propiedad:

```
animation-direction: alternate;
```

el valor "alternate" hace que la animación al llegar al final se repita en sentido inverso. También puede tener el valor `normal` que hace que la animación vaya sólo en una dirección.

NAVEGADORES: Al igual que la regla `@keyframes` las propiedades de animación han sido incorporadas recientemente en su forma estándar en todos los navegadores excepto Safari y Crhome, que siguen utilizando la forma experimental. Internet Explorer adopta la forma estándar a partir de su versión 10.

Ejemplo de animación:

Con lo que hemos visto hasta ahora ya podemos hacer un ejemplo de animación sencillo. El ejemplo consistirá en un cuadrado que se mueve horizontalmente mientras cambia de color.

veamos primero la regla `@keyframes`. Para verla en todos los navegadores la pondremos primero en la versión estándar, y luego en la forma para Safari y Chrome:

```
/*Regla keyframes en forma estándar*/
@keyframes mianim {
   from { left: 50px; background-color:red } /*Fotograma inicio*/
   to { left: 600px; background-color:blue } /*Fotograma final*/
}
/*Regla keyframes para Chrome y Safari*/
@-webkit-keyframes mianim {
   from { left: 50px; background-color:red } /*Fotograma inicio*/
   to { left: 600px; background-color:blue } /*Fotograma final*/
}
```

En los fotogramas clave marcamos en el de inicio la posición de partida (`left: 50px`) así como el color que tendrá cuando esté en esa posición (`background-color:red`); y en el fotograma final la posición y color que tendrá cuando llegue al final `left: 600px; background-color:blue`). Hemos usado aquí las palabras clave `from` para el inicio y `to` para el final, que equivalen al `0%` y `100%` respectivamente.

El elemento animado debe tener un **posicionamiento**, que puede ser relativo o absoluto. De esta forma podemos usar las propiedades de posicion (left, right, top, bottom) para cambiarlo de sitio y moverlo por la página. Además le daremos las propiedades de anchura, altura y borde para convertirlo en un cuadrado. Escribimos esto en el archivo CSS:

```
/*posicionamiento, anchura, altura y  borde elemento.*/
#anim { position: relative ;
        width: 100px; height: 100px;
        border: 1px solid black; }
```

Ahora pondremos las propiedades de animación necesarias para que la animación funcione, que son las indicadas anteriormente. Al igual que con la regla `@keyframes` debemos repetirlas en su forma experimental para que puedan verse en Chrome y Safari

```
/*propiedades de animación*/

#anim { /*Forma estándar*/
        animation-name: mianim; /*referencia nombre*/
        animation-duration: 5s; /*tiempo*/
        animation-iteration-count: infinite; /*repeticion indefinida*/
        animation-direction: alternate; /*repetir al revés*/
```

```
        /*Para Chrome y Safari*/
        -webkit-animation-name: mianim; /*referencia nombre*/
        -webkit-animation-duration: 5s; /*tiempo*/
        -webkit-animation-iteration-count: infinite; /*repeticion
indefinida*/
        -webkit-animation-direction: alternate; /*repetir al revés*/
      }
```

Guardamos el archivo CSS (*mianim.css*) y creamos el archivo HTML, el cual será así de sencillo:

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Mi Primera Animación</title>
  <link rel="stylesheet" type="text/css" href="mianim.css" />
</head>
<body>
  <h1>MI primera animación</h1>
  <div id="anim"></div>
</body>
</html>
```

Propiedades de animación

Antes de empezar con las propiedades de animación, veamos algo más sobre los fotogramas clave:

Fotogramas clave

Recordemos la regla "@keyframes";

```
@keyframes <nombre> { <fotograma> || <fotograma>+ }
```

El <nombre> es un elemento identificativo. Podemos poner cualquier palabra siempre que sigamos una serie de reglas que son las mismas que para escribir archivos, es decir, no utilizar espacios en blanco, no empezar por un número, se puede usar el guión normal o el guión bajo, y no es recomendable utilizar acentos o la letra "ñ".

Los "<fotograma>"; : Cada fotograma es en sí una regla CSS compuesta de un selector y una declaración, en la que puede haber una o varias propiedades. Es obligatorio poner al menos dos fotogramas, que serán el inicio y el final de la animación.

el selector del <fotograma> : El selector indica el punto temporal de la animación en el que se llega al estado marcado en las propiedades. Si sólo hay dos fotogramas los selectores deben marcar el inicio y final de la animación y pueden ser las palabras clave `from` (inicio) y `to` (final).

Si hay más de dos fotogramas debemos poner los fotogramas intermedios, para ello los selectores serán un número expresado en tantos por ciento; el `0%` Es el fotograma de inicio, y es obligatorio, el `100%` es el fotograma final, y también es obligatorio. Si sólo hay dos fotogramas también podemos expresarlo así, pero si hay más pondremos como selector los tantos por ciento intermedios de cada fotograma.

Las propiedades del <fotograma> : Dentro de la declaración del fotograma (entre llaves) pondremos las propiedades que queremos que cambien a lo largo de la animación. Lo normal es poner las mismas propiedades en varios fotogramas, pero con diferentes valores.

El paso de un fotograma clave al siguiente se realiza de forma progresiva, siempre que esto sea posible. Sólo las propiedades cuyo valor esté basado en un número (números, medidas, porcentajes, colores) pueden variar progresivamente. Las propiedades cuyos valores no son números (palabras clave, nombres, urls, etc.) son ignoradas, y no se puede hacer ningún tipo de cambio en ellas.

Para mover un elemento dentro de la página, éste debe estar posicionado, es decir, tiene que tener la propiedad `position: absolute;` o `position: relative;`, ya que de otro modo no se le pueden aplicar las propiedades de posicionamiento (`left`, `right`, `top`, `bottom`), que son las que cambian el elemento de sitio.

Propiedad animation-name.

La propiedad `animation-name` es obligatoria o imprescindible para que la animación funcione correctamente, ya que indica a qué elemento deben aplicársele los fotogramas clave de la regla `@keyframes`:

```
#capa1 {animation-name: <nombre>;}
```

El valor (`<nombre>`) debe ser el <nombre> que se ha puesto en el selector de la regla `@keyframes`. Así los fotogramas se aplicarán al elemento al que se refiere esta regla (`#capa1`).

Propiedad animation-duration:

La propiedad `animation-duration` indica el tiempo que va a durar la animación. Es por tanto imprescindible u obligatoria, ya que toda animación tiene siempre una duración en el tiempo:

```
animation-duration: <tiempo>;
```

Los valores de `<tiempo>` se expresan siempre mediante un número seguido de la letra "s" cuando el tiempo es en segundos, o "ms" cuando es en milisegundos.

Propiedad animation-direction:

La propiedad `animation-direction` indica si la animación al llegar al final acaba, o si debe volver a verse en sentido inverso:

```
animation-direction: normal | alternate;
```

Los dos posibles valores que tiene esta propiedad son:

- `normal` : Es el valor por defecto, la animación, una vez concluido el tiempo que se le ha marcado finaliza.
- `alternate` : Este valor indica que una vez finalizada la animación esta se repite en sentido inverso, volviendo a reproducirse la secuencia de atrás a adelante.

Tanto esta propiedad como las siguientes no son obligatorias, sin embargo la mayoría de las veces debemos ponerlas para conseguir los efectos deseados.

Propiedad animation-iteration-count

La propiedad `animation-iteration-count` indica las veces que se repetirá la animación, éstas pueden ir desde 1 hasta infinito.

```
animation-iteration-count: <entero_positivo> | infinite;
```

Los posibles valores son la palabra clave `infinite` que indica que la animación se repetirá indefinidamente, o un número entero positivo, que indica el número de veces que se repetirá la animación. Su valor por defecto es 1, es decir, de no poner esta propiedad la animación se ejecuta una vez.

Las propiedades vistas hasta aquí son las que hemos empleado en la página anterior para realizar nuestro ejemplo de "mi primera animación", pero hay más propiedades:

Propiedad animation-delay

La propiedad `animation-delay` indica el tiempo que tarda la animación en empezar después de cargarse la página:

```
animation-delay: <tiempo>;
```

El valor es una unidad de tiempo que se expresa en segundos o milisegundos, igual que en la propiedad `animation-duration`. Tras cargarse la página, se esperará el tiempo indicado para iniciar la animación.

Propiedad animation-timing-function

La propiedad `animation.timing-funtion` controla los cambios en la velocidad mientras se produce la animación.

```
animation-timing-function: linear | ease | ease-in | ease-out |
ease-in-out | cubic-bezier(n,n,n,n);
```

La velocidad a lo largo de la animación puede sufrir cambios, los diferentes valores de esta propiedad controlan la velocidad a lo largo de la animación. Éstos son:

- `linear` : La velocidad de la animación se mantiene constante durante toda su duración.
- `ease` : Es el valor por defecto. La animación va lenta al principio para volverse rápida en el medio, y terminar mucho más lenta.
- `ease-in` : La animación se muestra lenta al principio, para coger al final su máxima velocidad.
- `ease-out` : La animación es rápida al principio y va disminuyendo en velocidad para ser lenta al final.
- `ease-in-out` : La animación va lenta al principio, se vuelve rápida a la mitad, para acabar lenta.
- `cubic-bezier (n,n,n,n)` : Definimos la velocidad de la animación en base a una curva cúbica de Bezier. Para ello incorporamos cuatro valores "n" que serán números decimales entre el 0 y el 1.

Las curvas de Bezier son el resultado de unas fórmulas matemáticas, si no estás familiarizado con esto puedes consultarlo en Wikipedia: curvas de Bézier; y en concreto el apartado Construcción de curvas de Bézier, donde se muestra mediante gráficos cómo se construyen estas curvas.

Propiedad animation tipo "shorthand"

Todas las propiedades anteriores de animación pueden reunirse en una sola de tipo "shorthand":

```
animation:<animation-name> || <animation-duration> || <animation-
timing-function> || <animation-delay> || <animation-iteration-count>
|| <animation-direction>]
```

Como valores pondremos primero el nombre `animation-name`, después el tiempo de duración: `animation-duration`; después el tipo de velocidad `animation-timing-function`, seguimos con el tiempo de retardo de comienzo `animation-delay`, después el número de repeticiones `animation-iteration-coun` y acabamos con la propiedad de dirección `animation-direction`.

Este es el orden recomendado por la W3C. Las propiedades que no son imprescindibles pueden omitirse. El orden de las propiedades puede cambiar en las propiedades no obligatiorias. Las propiedades `animation-name` y `animation-duration` (obligatorias) deben ser las primeras.