UD3 HTML Y CSS.

DISEÑO DE INTERFACES WEB

3.5 Animaciones CSS3

Contenidos

1. Objetivos

En este quinto tema se pretenden conseguir los siguientes objetivos:

- Crear animaciones sin necesidad de programar utilizando las animaciones de CSS3.
- Controlar la reproducción de las animaciones CSS3 mediante JavaScript. Encadenar animaciones CSS3.

2. Introducción

En temas anteriores vimos cómo crear transiciones sobre elementos de nuestra página. Veremos en este tema, cómo encadenar estas transiciones utilizando las animaciones CSS3.

Además, veremos cómo podemos controlar la reproducción de estas animaciones y encadenar una tras otra, utilizando para ello los nuevos eventos para animaciones incorporados al lenguaje JavaScript.

4. Animaciones en 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.

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.

4. Animaciones en CSS3

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

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

```
@keyframes animacionCoche
{
      /*Indicamos que salimos de la posición 0*/
      from     0 %
      {
            left:0px;
      }
      /*Indicamos que al final la posición debe ser 350*/
      to     100 %
      {
            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, se debe especificar estos **dos fotogramas** para que el navegador sepa dónde comenzar y finalizar; debido a su importancia, estos fotogramas tienen alias especiales: **from y to**.

@keyframes

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;
    }
}
```

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:

Asignar animaciones CSS3 a un elemento

- `animation-delay`: tiempo en segundos entre que se carga el elemento y el comienzo de la animación..
- `animation-direction`: indica la dirección de la animación. Los posibles valores son:
  - normal: Se reproduce desde el inicio hasta el final.
  - reverse: 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.
- `animation-name`: Especifica el nombre de la regla @keyframes que describe los fotogramas de la animación.

Asignar animaciones CSS3 a un elemento

- `animation-play-state`: permite pausar y reanudar de la animación. Se podría usar desde JS para pausar la animación.
- `animation-timing-function`: indica el ritmo de la animación estableciendo curvas de aceleración. Posibles valores: ease (por defecto), 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.

Asignar animaciones CSS3 a un elemento

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;    --> Es el  @keyframe
    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.

Asignar animaciones CSS3 a un elemento

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

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

    position:relative;
}
```

Controlar la reproducción de las animaciones

También nos puede interesar reproducir una animación como **respuesta a un determinado evento**. Por ejemplo, que se inicie cuando colocamos el ratón sobre un elemento o al hacer click.

```
#coche          #coche:hover{       #coche:active
{               {                   {
```

Este caso es bastante obvio y no supone ninguna complicación, pero ¿qué sucedería si quisiéramos que una animación se iniciara cuando hacemos **click sobre otro 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; }
}
```

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

   position:relative;
}
```

Controlar la reproducción de las animaciones

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, 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);
```

Cuando pulsemos sobre el botón, se le **asignará la clase aCoche al elemento coche**, y en este momento se iniciará la animación.

```
/* para repetir sin tener que recargar, limpiamos el evento */
coche.addEventListener('animationend',()=>{
   coche.classList.remove("aCoche")
   }
)
```

Encadenar animaciones

Para **encadenar animaciones y que una se inicie cuando termine la anterior**, disponemos de tres eventos que nos darán información sobre el estado en el que se encuentra la animación.

- `animationstart`: este evento será lanzado cuando se inicie la animación **por primera vez**. Si la animación está configurada para hacer varias iteraciones, solo 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.

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. En este ejemplo obviaremos el código para la compatibilidad con navegadores.

Encadenar animaciones

Código JavaScript para encadenar 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.

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

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

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