

30 minutos buscando mi TAG. +

¿Te gusta el diseño web? ¡Echa un vistazo a la docume

Recibe las actualizaciones de  
**Emezeta.com** en tu correo:

m!



Twitter!

14 11min

Leer otros artículos

# Animar personajes con animaciones CSS

¿Sabías que hay una forma muy sencilla de crear animaciones con sprites en CSS? En esta guía o tutorial te explico cómo crearlas.

Publicidad

Hace años que el [lenguaje CSS](#) (propuesto por [Håkon Wium Lie](#), uno de los desarrolladores de Opera) evolucionó de un sencillo y simple **lenguaje de marcado** a una completa **especificación**. Partió con el objetivo de encargarse de los **aspectos visuales de diseño de un documento web**, pero ha acabado siendo capaz de controlar mucho más: generación de contenido virtual, responsive web design y media queries, tipografías externas al sistema, transiciones, animaciones, transformaciones 2D y 3D, etc... Y precisamente,

de una de estas últimas es de las que vamos a hablar en este artículo: **Las animaciones CSS**.



# Guía para animar personajes con **Animaciones CSS**

EMEZETACOM

CSS permite crear sencillas o complejas **animaciones** en apenas algunas líneas de código, permitiendo modificarlas fácilmente (*e incluso automatizarlo mediante Javascript*) y convirtiéndolo en una herramienta extremadamente útil y flexible. En esta guía nos va a acompañar nuestro amigo **Bernard Bernoulli** (*Maniac Mansion, El día del Tentáculo*), que nos mostrará como se hacen algunos ejemplos sencillos. Puedes abrir una nueva pestaña de [CodePen](#) e ir pegando el código HTML/CSS e ir viendo como funciona.

## 1. Preparación de la región del personaje

Para empezar, la parte del código **HTML** será extremadamente sencilla. Simplemente especificamos una capa **div** con id **bern** (*de Bernard*) para hacer referencia a él posteriormente:

```
<div id="bern"></div>
```

Comenzamos con el **CSS**. En la capa que acabamos de mencionar vamos a colocar nuestro personaje animado. Quizás, lo más complicado del asunto es conocer las dimensiones de cada viñeta (*y que estén bien colocadas*), así que hay que ser cuidadoso con este asunto. Herramientas interesantes para la generación o manipulación de sprites, pueden ser [SpriteCow](#) o [SpritePad](#). Además, también tenemos [SpriteDatabase](#), una completísima base de datos con diferentes sprites de videojuegos que podemos utilizar para estos ejemplos. Si tienes tiempo, interés y buena mano, incluso puedes seguir esta guía y [crear tu propio personaje de 8 bits](#).

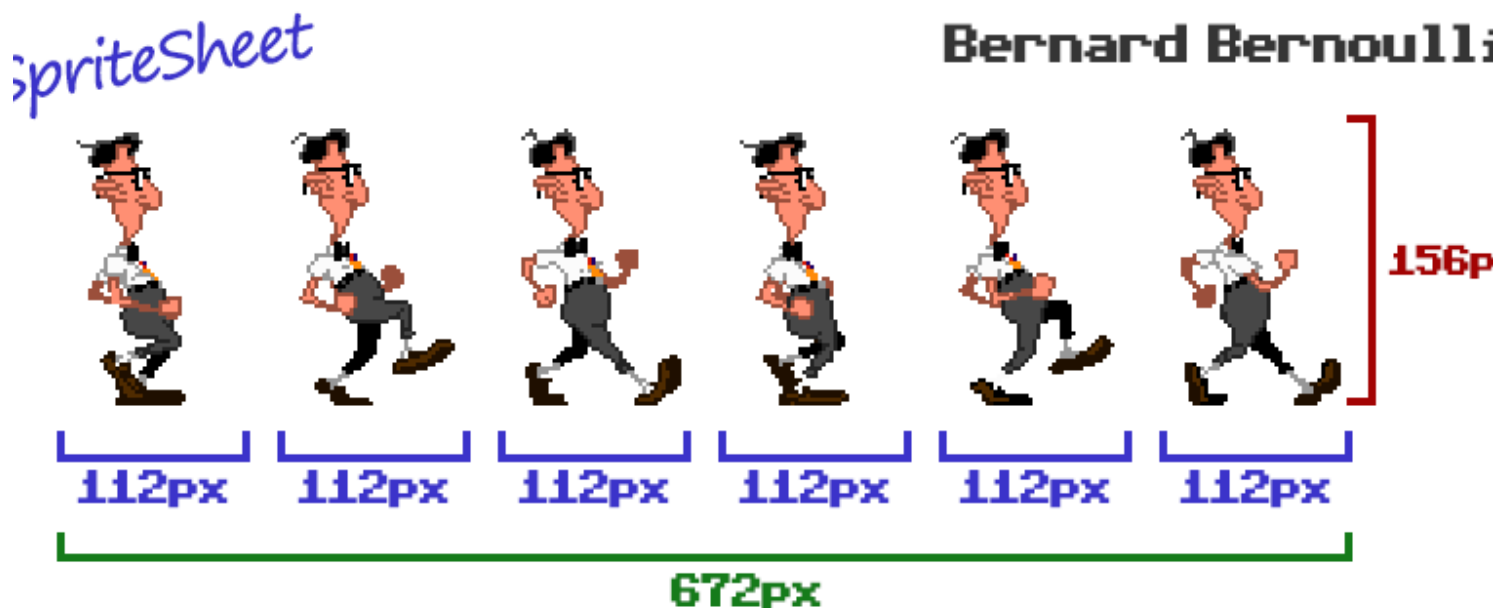
De momento, nosotros nos centraremos en definir el tamaño de la viñeta a **112x156** píxeles (*más adelante veremos por qué*) y le ponemos un fondo rojo para guiarnos y ser conscientes desde el principio de las dimensiones de esta región.

**Nota:** Ojo también a esto. Al establecer un **margen automático** (*margin:auto*) y un **ancho definido** (*width:112px*), lo que estamos haciendo es decirle al navegador que establezca dos márgenes (*izquierdo y derecho*) del mismo tamaño. ¿El resultado? Objeto centrado horizontalmente. Veamos todo lo que hemos hecho en código CSS:

```
#bern {  
  width:112px;  
  height:156px;  
  margin:auto;  
  background:red; /* Temporal, sólo para guiarnos */  
}
```

## 2. Preparación del sprite sheet animado

Una vez hecho esto, veremos que aparece un recuadro rojo centrado en pantalla. Esta región es donde colocaremos al personaje animado, utilizando lo que se llama un **Sprite Sheet** (*una imagen con todos los frames o viñetas de la animación del personaje*). La que nosotros utilizaremos en este caso es la [animación caminando de Bernard](#) ➔. Una pequeña explicación rápida:



Sprite sheet de Bernard (El día del tentáculo)

Vemos que cada viñeta del personaje ocupa exactamente el mismo tamaño de ancho (112px), mientras que el ancho total del sprite sheet es de 672px.

Publicidad

---

Volvemos al código y reemplazamos la propiedad **background** (*que sólo tiene el color rojo de fondo*) por la mencionada imagen. Aunque no es necesario (*porque actúa así por defecto*), especificamos **repeat-x** para hacer que la imagen se repita horizontalmente de forma indefinida:

```
background: url(http://i.imgur.com/ifk0SLH.png) repeat-x;
```

En principio nos aparecerá sólo la primera imagen (viñeta) de Bernard, ¿Por qué? Porque hemos especificado la imagen y la hemos «recortado» con las propiedades **width** y **height**. Si eliminamos temporalmente la propiedad **width**, veremos que nos aparecen todas las viñetas de Benard, repitiéndose hasta el infinito. Esto ocurriría así porque no habríamos limitado la región visible del personaje.

Pero volvamos al caso anterior. Nos aparece sólo la primera viñeta de Bernard. Esto ocurre porque la imagen de fondo se coloca en la posición 0 de la región del personaje, es decir es equivalente a añadir este código CSS:

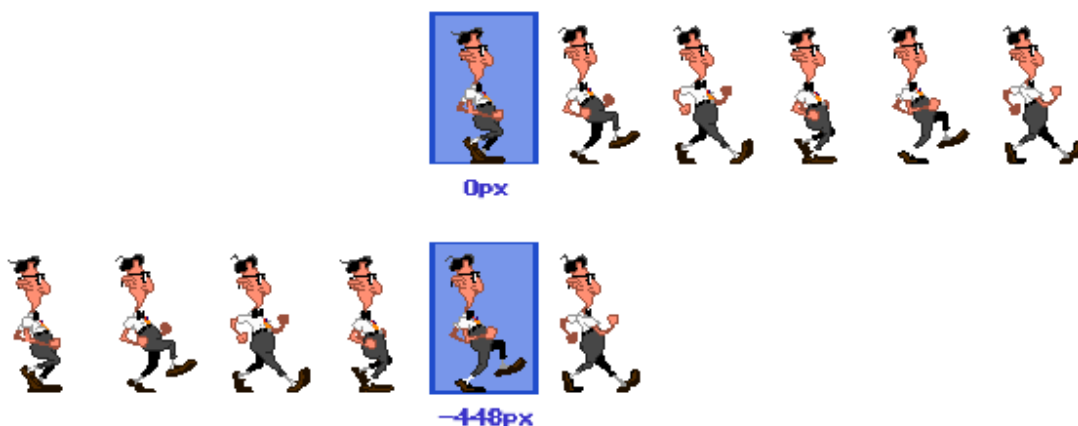
```
background-position:0;
```

De hecho, si modificamos esta propiedad **background-position:0** por **background-position:-112px**, veremos que en lugar de la primera, nos aparece la segunda viñeta:

```
background-position:-112px;
```

Si somos conscientes de que la imagen completa ocupa **672 píxels de ancho** y que cada viñeta ocupa **112 píxels de ancho**, es lógico pensar que la primera viñeta empieza en el píxel 0, la segunda en el píxel 112, la tercera en el píxel 224 y así sucesivamente. Veamos gráficamente lo que está ocurriendo:

**background-position:**



Funcionamiento de background-position en un sprite CSS animado

Realmente, lo que estamos modificando con **background-position** es, como su nombre indica, la posición de la imagen de fondo, de modo que si le indicamos **112 píxeles negativos**, significa que va a mover la imagen **112 píxeles** a la izquierda, ubicando la siguiente viñeta «bajo» la región del personaje (*que no se ha movido, realmente*). Si en lugar de un **valor negativo** pusieramos un valor positivo, veríamos que recorreremos las viñetas en orden inverso (*podría estar bien para un «moonwalk» con un sprite de Michael Jackson*).

### 3. Animar la imagen del personaje

Una vez entendido esto, eliminamos la propiedad **background-position** anterior (*que sólo habíamos utilizado para comprender lo que estaba ocurriendo*) y nos centramos en definir la animación del personaje caminando, que es la que llevará dicha propiedad. Para ello, tenemos que crear un bloque **@keyframes**:

```
@keyframes walk {  
  0% { background-position:0 }  
  100% { background-position:-672px }  
}
```

En este bloque lo que hacemos es definir una animación **walk** (*caminar*) que va a partir de un primer fotograma (**0%**) con la posición de la imagen de fondo a **0 píxeles** (*inicio de la primera viñeta de la imagen*) y va a terminar en el último fotograma (**100%**) con la posición de la imagen de fondo a **-672 píxeles** (*final de la última viñeta de la imagen*). De esta forma, CSS se encargará de interpolar (*inventarse*) toda la animación para que parta de ese estado inicial a ese otro estado final.

Pero nos queda una última parte importante. Hemos definido la animación pero no la hemos «enlazado» con ningún elemento HTML, por lo que hasta ahora no le hemos dicho a CSS donde aplicar la animación. Para enlazarla, lo único que hacemos es incluir la propiedad **animation** dentro del bloque **#bern** donde estabamos aplicando estilos:

```
animation:walk 8s linear infinite;
```

De esta forma, le estamos diciendo que al elemento con id **bern** le aplique la animación **walk** que debe durar **8 segundos** (*desde el primer fotograma, 0%, hasta el último, 100%*), que utilizará una función de tiempo lineal (*cada fotograma tiene la misma duración*) y que se repetirá indefinidamente.



## Animación lineal

Emezeta.com

Con una animación lineal, la animación se produce de forma constante, con la misma duración entre cada fotograma y sin saltos.

### Animación CSS lineal

**Nota:** Para una mayor compatibilidad de navegadores, en la ventana de CSS de CodePen, hay que pulsar el icono de la rueda dentada y seleccionar **Autoprefixer** en la penúltima opción. Esto hará que el código CSS escrito pase por [autoprefixer](#) y añada los prefijos necesarios.



## 4. Utilizar `steps()` para saltar fotogramas

Sin embargo, esto dista mucho de lo que queremos hacer. Simplemente he definido una primera **animación lineal** para que se entienda como se aplica. Ahora, lo que debemos hacer es reemplazar el **linear** del código anterior por **`steps(6)`** que es una **función de tiempo especial** que divide la imagen en los fragmentos que le indiquemos (*en nuestro caso 6*).

Como tenemos los cálculos hechos correctamente,  $672/6 = 112$ , por lo que si el sprite sheet está bien diseñado la animación debería ser perfecta, realizando saltos de esta manera:



Avance de una animación CSS usando `steps()`

Lo único que tenemos que hacer es ajustar los **8s** que especificamos antes y reducirlos a gusto del consumidor, **1s** por ejemplo. Realmente, la animación no ocurre de esta forma, sino que recordemos que lo que estamos modificando es el **background-position**, por lo que la animación se realiza moviendo la posición de la imagen de fondo, de la siguiente forma:



Efecto de la animación de background-position

¡Perfecto! En apenas 10 líneas de código CSS (y *una de HTML*) ya tenemos una animación funcionando. Pero vamos a darle un poco más de contexto a la animación.

## 5. Establecer una imagen de fondo

Una buena idea sería ponerle una **imagen de fondo** a la animación, aprovechando que los sprites del personaje están en formato PNG y tienen fondo transparente (*si quieres más información sobre este tema, echa un vistazo a la [guía de formatos de imágenes](#)*). He extraído la siguiente imagen del muelle de [Monkey Island](#). Podríamos usarla tal cual, pero para darle algo más de dinamismo, la he recortado en dos fragmentos: [el cielo](#) y [el mar](#).

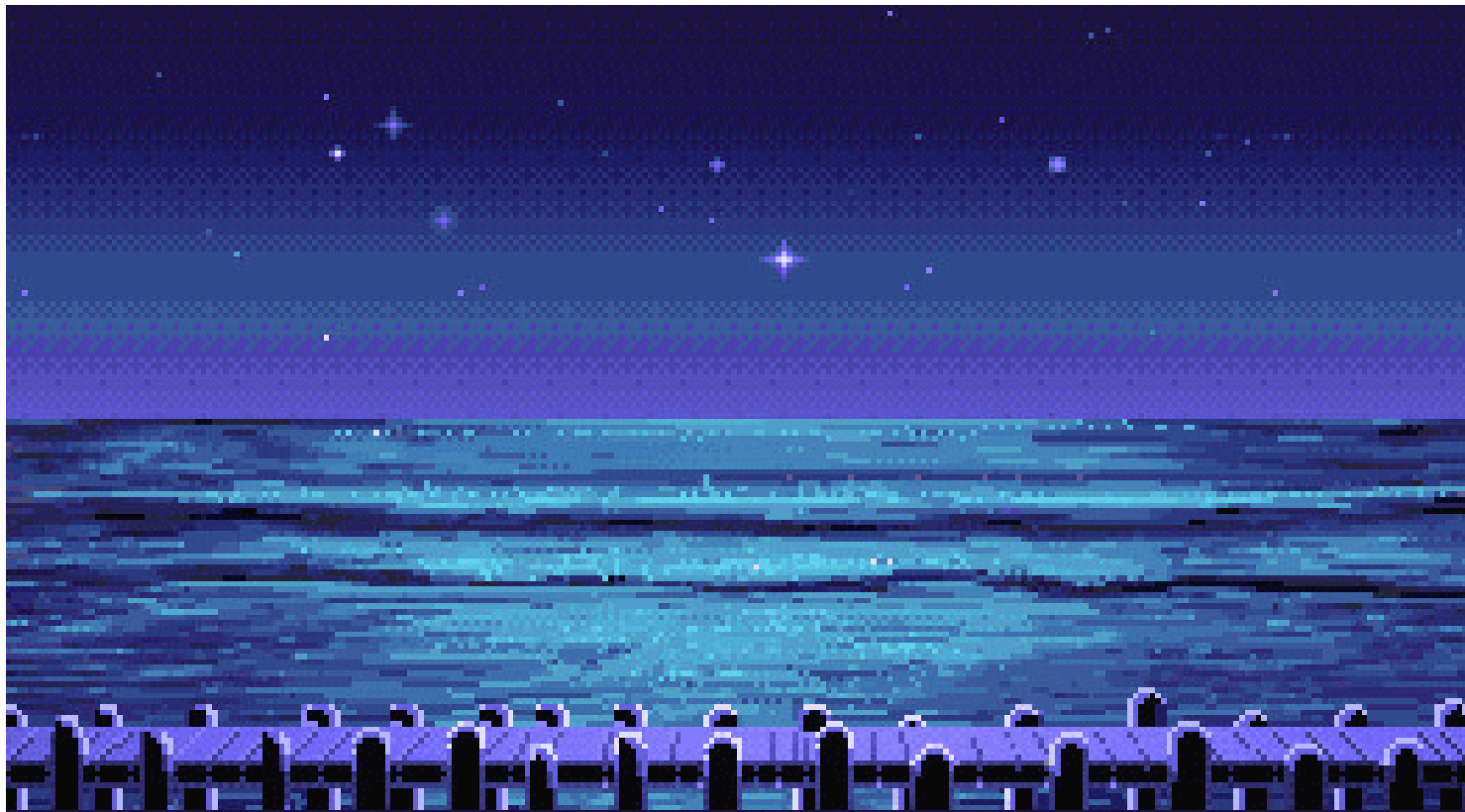


Imagen de fondo de Monkey Island

Lo primero que haremos es modificar el código HTML para añadir estos dos componentes **#sky** (*cielo*) y **#sea** (*mar*). Los anidaremos uno dentro de otro para que sea más sencillo mover los elementos después:

```
<div id="sky">
  <div id="sea">
    <div id="bern"></div>
  </div>
</div>
```

Luego, en el código CSS haremos también algunos añadidos. En primer lugar, definimos el cielo con la propiedad **background**:

```
#sky {
  background:url(http://i.imgur.com/PhHVjgw.png) repeat-X;
```

```
}
```

Y en segundo lugar, definimos el mar. Este último, además, veremos que se solapa con la imagen del cielo, por lo que debemos posicionarlo desde la parte superior (*top*) **145 píxels** hacia abajo:

```
#sea {  
  background:url(http://i.imgur.com/h75XWy8.png) repeat-x;  
  position:relative;  
  top:145px;  
}
```

Bien, ya está hecho. Sin embargo, comprobaremos que **Bernard** se queda caminando en la parte inferior, por lo que tendremos que posicionarlo desde la parte inferior (*bottom*) un poco hacia arriba, incluyendo el siguiente código CSS en **#bern**:

```
position:relative;  
bottom:34px;
```

## 6. Animar el fondo (Parallax)

Ya tenemos un fondo colocado. Sin embargo, un fondo estático queda un tanto raro, ya que parece que **Bernard** este realizando alguna danza de la victoria al estilo de **Will Smith**. Para ello, vamos a aplicar también una animación al fondo. Primero definimos dicha animación:

```
@keyframes movebg {  
  0% { background-position:550px }  
  100% { background-position:0 }  
}
```

La animación **movebg** moverá las imágenes de fondo (*cielo y mar, que tienen el mismo ancho*) con un **background-position** desde 550px hasta 0. Si se fijan, aquí lo hemos hecho al revés, desde el final de la imagen hacia el inicio, por eso no usamos valores negativos (*otra forma podría ser de 0 a -550px*). Aquí si se trata de una animación lineal, ya que no tiene sentido utilizar saltos con `steps()` en este caso.

Y como ya habíamos visto antes, faltaría enlazar la animación a los elementos HTML. Así pues, incluimos el siguiente código CSS a los elementos **#sky** y **#sea**:

```
animation: movebg 9s linear infinite;
```

Sin embargo, en la línea anterior de **#sea** modificaremos el tiempo de duración de la animación de **9s** a **6s**. De esta forma, conseguiremos un [efecto Parallax W](#) en el que simulamos profundidad, ya que cada fondo va a una velocidad diferente.

Finalmente, podemos añadir el siguiente fragmento de código CSS para eliminar los márgenes del documento y cambiar el fondo blanco a un fondo negro más elegante:

```
body {  
  margin:0;  
  background:#000;  
}
```

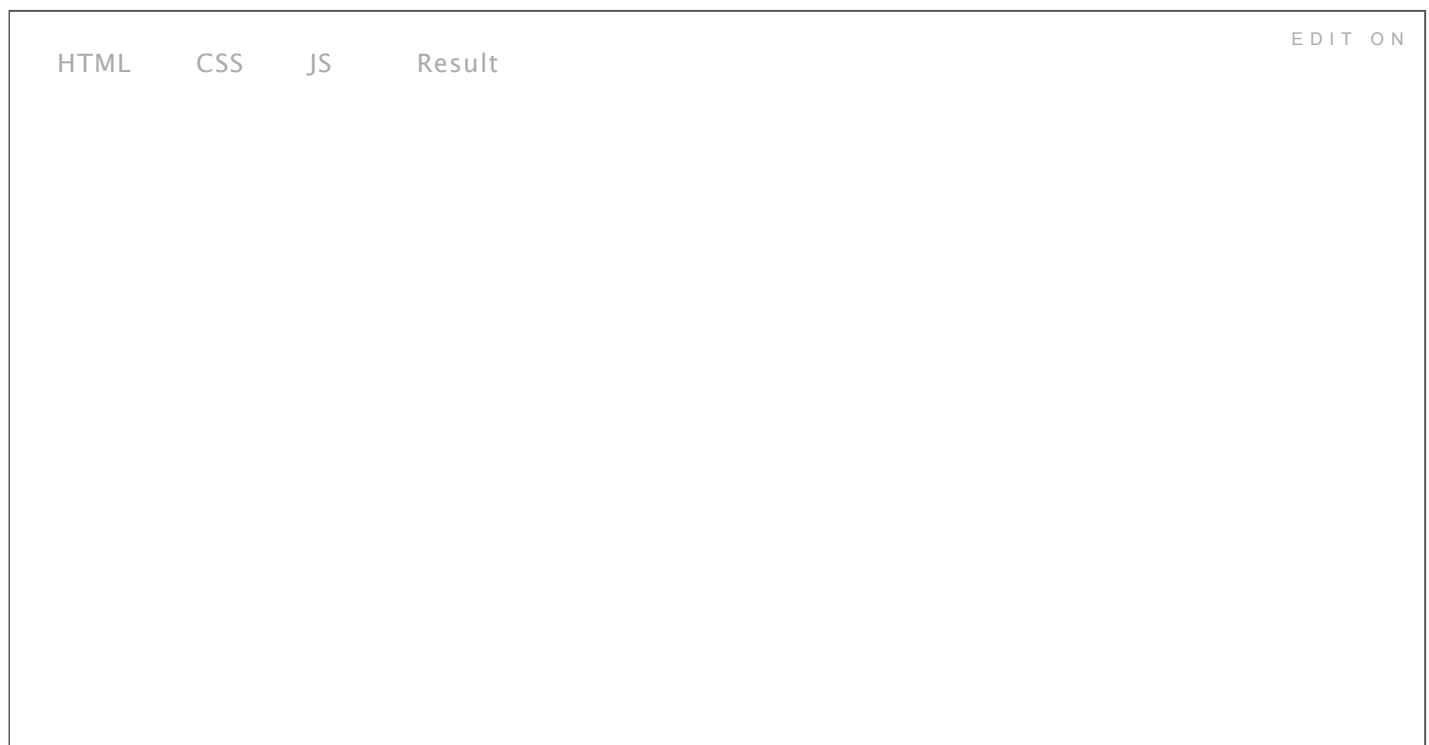
## 7. Música de fondo

Y si queremos rizar el rizo, podemos añadir un pequeño fragmento HTML para aderezar la animación con una música de fondo que acompañe a

Bernard:

```
<audio id="song" autoplay>
  <source src="http://scummbar.com/mi2/DOTT/03%20-
%20Busting%20The%20Candy%20Machine.mp3">
</audio>
```

¡Listo! Puedes pulsar sobre **Edit on CodePen** para modificar el código o hacer un fork (*le he retirado el autoplay para no molestar durante la lectura de este artículo*):



## 8. Opcional: Encadenar animaciones

Aunque no se muestra en este tutorial, algo interesante que puede hacerse es **encadenar múltiples animaciones**, pudiendo realizar animaciones finitas. En el ejemplo anterior, he realizado siempre animaciones usando «infinite», es decir, que se repitan siempre. Sin embargo, se puede reemplazar por un **número** que indica el número de veces que se realiza una animación. En caso

de omitirlo, la animación se realizará una sola vez. También es posible establecer retardos, algo muy interesante en animaciones encadenadas. Por ejemplo:

```
animation: moveright 5s linear,  
          moveleft 5s linear 5s,  
          disappear 6s linear 10s;
```

En este fragmento de código, se está realizando una animación múltiple compuesta por 3 animaciones encadenadas:



### Animaciones CSS múltiples (animaciones encadenadas)

- La primera línea aplica la animación **moveright**, donde mueve hacia la derecha un personaje durante 5 segundos. Como no tiene definido *infinite* la animación se realiza una sola vez y no se repite. No tiene retardo (*es la primera en ocurrir*).
- La segunda línea aplica la animación **moveleft**, donde mueve hacia la izquierda un personaje durante 5 segundos. Tampoco tiene *infinite* por lo

que también se realiza una sola vez. Pero si nos fijamos, tiene un segundo valor **5s** esto es el retraso que tendrá dicha animación antes de realizarse (*lo que dura la primera*).

- La tercera línea aplica la animación **disappear**, donde hace desaparecer un personaje utilizando **opacity** durante 6 segundos. En este caso, el retardo es de 10 segundos (*lo que dura la suma de la primera y la segunda animación*).

De esta forma tenemos un mecanismo interesante para realizar animaciones finitas compuestas por varias animaciones sucesivas.

## 9. Opcional: Algunas posibles mejoras

Comento algunos detalles que se podrían mejorar sobre el ejemplo de este artículo:

- **Más capas:** En mi caso, he utilizado sólo dos capas (*cielo y mar*) para el efecto Parallax. Se podrían utilizar más capas a diferentes velocidades o incluso objetos como la luna o algunas maderas en primer plano para simular el efecto de avance.
- **Seamless:** En la imagen del mar de fondo se nota muchísimo el corte de la misma. Hay técnicas para disimular este recorte, que pueden encontrarse en Internet buscando por la palabra clave [seamless](#) ➡.
- **Javascript:** Es posible utilizar JS/jQuery para alterar ciertos detalles de la animación al interactuar con el usuario, como por ejemplo detener la animación con la propiedad **animation-play-state:paused** o añadiendo una clase **.rev** a **#bern** que utilice **transform:rotateY(-180deg)** y le de la



vuelta al personaje para que camine hacia el otro lado. Incluso manejar al personaje con los cursores del teclado, aprovechando todo lo anterior.

En el código de CodePen hay algunas líneas que pueden descomentarse y ver como funcionan, como sombras adicionales sobre las imágenes, por ejemplo.

## 10. Extra: ¿Dónde está Guybrush?

Y por último, cómo sé que muchos me dirán en los comentarios «¡Eh! ¡Qué ese fondo que has utilizado no es de The Day of The Tentacle sino de Monkey Island!»...

URL | <http://codepen.io/manz/pen/wBZvoE> ➡

Recuerda que en [LenguajeCSS](#) ➡ tienes una **Chuleta CSS3** disponible para descargar (*sujeta a donación*), una extensa sección de documentación, enlaces y herramientas, y si eres de Tenerife, aún puedes apuntarte al siguiente [curso de CSS3](#) ➡ que imparto próximamente. ¿Alguien se anima a hacer su propia **animación CSS con sprite sheets** en CodePen y colocarla en los comentarios?

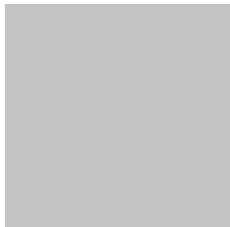
Escrito por [Manz](#), el Jueves 2 de abril de 2015, en [programación](#). Comentarios recibidos: **14**.

Este artículo aún no ha sido enviado a Menéame. Puedes [ser el primero en enviarlo](#) ➡!





 3  14,9K

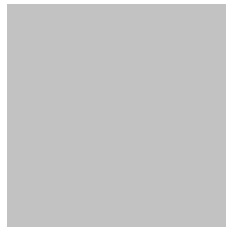
## Niveles de error en PHP



Una aplicación o código en lenguaje PHP tiene varios niveles de error (`error_reporting`). En este breve artículo vamos a hablar del `error_reporting` y sus características.

 39  479K

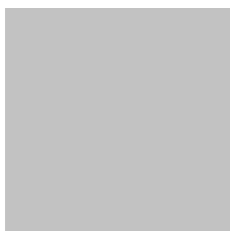
## ¿Cuánta memoria RAM consume mi servidor?



¿Cuánta memoria RAM consume mi servidor por petición? Una buena forma de medir el consumo de tu servidor web es calculando el gasto por petición y teniendo en cuenta las páginas vistas.

 21  23,4K

## La jerarquía del programador



Basado en el clásico The Programmer Hierarchy de Lukewelling.com, presentamos infografía del diagrama de La jerarquía del programador.

[Leer más artículos](#)

---

Artículo escrito por **J. Román Hernández**, más conocido como **Manz** (autor de Emezeta). Es



Ingeniero-técnico informático de Gestión por la Universidad de La Laguna y reside en Santa Cruz de Tenerife (Canarias). Puedes seguirlo en las siguientes redes:

 Twitter  LinkedIn  Google+

---

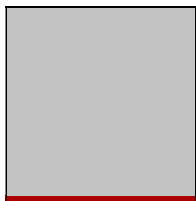
### Anuncios patrocinados



## 14 comentarios de lectores

Publicidad

---

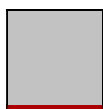
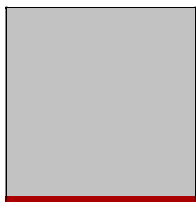
**KATREyuk**

Lunes, 6 de abril de 2015, 11:44

Qué pasada :)

Curioso ver a Bernard sobre el escenario del Monkey... qué grandes!

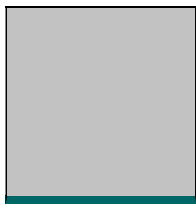
Gracias Manz

**Hernán Espectacular****Sebas**

Jueves, 7 de mayo de 2015, 21:40

¿Existirá la forma de, por ejemplo, que empiece la animación de caminar en el 3 frame y de ahí siga haciendo el ciclo entero normal?.



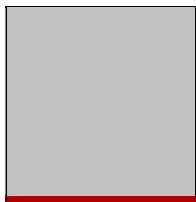


Autor

**Manz**Sábado, 9 de mayo de 2015, 14:13

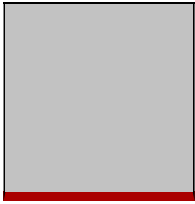
[@Sebas](#) ➡ : Muy buena pregunta. En la propiedad **animation-delay** se pueden poner valores negativos para conseguir lo que buscas. Por ejemplo, si tenemos una animación con **animation-duration: 10s** y establecemos **animation-delay: -5s**, la animación empezará por la mitad.

Saludos,

**Lucas Plus**Jueves, 21 de mayo de 2015, 09:46

Buenos consejos, aunque resulta un verdadero quebradero de cabeza, a ver que encuentro por las redes... Me encanta los ejemplos. Un saludo!

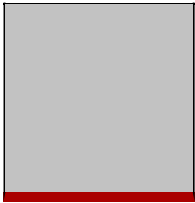




## Marcelo

Sábado, 19 de septiembre de 2015, 10:24

Excelente artículo. Felicitaciones y gracias x compartirlo. Saludos desde Belgica !



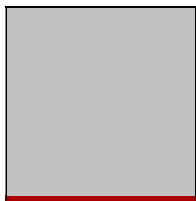
## Joksan

Martes, 27 de octubre de 2015, 17:59

Yo tengo una duda más puntual, haber si alguien sabe la respuesta, ¿es posible encadenar/asociar una animación, transición SVG o CSS, a un archivo multimedia? De modo que si yo tengo un mp3 la animación inicie en 4:32 min (por dar un ejemplo), de modo que si yo pauso el audio la animación no inicia hasta que se reanude y alcance el momento indicado. Algo así como un subtítulo.

He querido experimentar con animación tipográfica, pero si asigno los tiempos de forma manual sin ninguna tipo de sincronización, dependiendo de la velocidad de la conexión, los elementos pueden tardar en cargar y terminar desfasados.





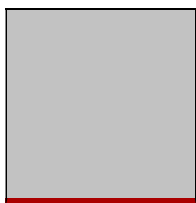
## Joksan

Miércoles, 17 de febrero de 2016, 19:02

ya encontré como: <http://svg-wow.org/blog/2009/10/04/animated-lyrics/>



+2

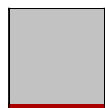


## Darktower

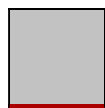
Lunes, 28 de marzo de 2016, 22:33

Como se haría con un Sprite sin dimensiones fijas entre imágenes si no que maneja anchos diferentes entre imágenes,

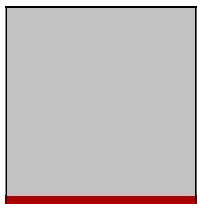
Muchas gracias por la guía esta majestuosa.



**RoyMora** Excelente muy bien explicado y muy util

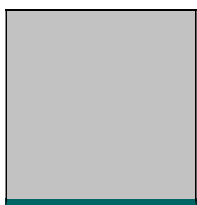


**RoyMora** muchas gracias excelente y muy util

**Jaime**

Jueves, 2 de febrero de 2017, 20:19

Hola que tal, yo tengo una duda con cómo encadenar animaciones, de qué forma declaraste el @keyframe con diferentes nombres de animación, o bien, serían varios @keyframe declarados?



Autor

**Manz**

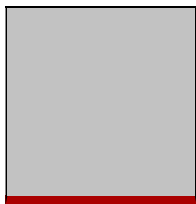
Jueves, 2 de febrero de 2017, 22:21

[@Jaime](#) ➡ : Primero se declaran los keyframes con las acciones genéricas a realizar, luego, se crea una sola propiedad animation, donde se van colocando las diferentes acciones (separadas por comas) y aprovechando la duración y el retardo de cada acción para ir encadenando una con otra.

¡Saludos!







## Angelica

Martes, 2 de mayo de 2017, 00:10

Hola que tal!, Saludos... Oye, disculpa, necesito hacer una animación parecida a la que tu realizaste en esta publicación, solo que necesito usar una imagen que yo tengo...

Es un logo, pero me piden que lo anime y seguí los pasos pero no creo que haya hecho todo bien porque ni siquiera me aparece la imagen  
Crees que podrías ayudarme? Muchas graciaas!



## Publica tu opinión

Si lo deseas, puedes utilizar el siguiente formulario para publicar tu opinión o responder a alguna de las existentes:

Tu nombre
tu@email.com

http:// (o perfil de Twit

Escribe aquí tu comentario... ¡Separa en párrafos los textos muy abundantes y revisa la previsualización del comentario antes de enviarlo! Tu comentario puede tardar algunos segundos en aparecer después de enviarlo.

- ☐ Acepto las [condiciones y políticas de privacidad](#) de este sitio web.
- ☐ Suscribirme a través de [FeedBurner](#) a los **nuevos artículos** del blog por email.

Publicar comentario

## Previsualización

Aquí se previsualizará su comentario. Revise que sea correcto antes de publicarlo.

## Artículos populares

## Las 100 mejores canciones de los 80



Es imposible reunir las mejores canciones de los 80, pero aquí va -a mi criterio- la lista de los mayores éxitos de la mejor década en el mundo de la música: los 80.

## 15 aplicaciones gratis para recuperar archivos borrados



Selección de 15 programas gratis para Windows que permiten recuperar información eliminada o borrada de nuestros discos o memorias.

## 13 paradojas que quizás no conocías



Una paradoja es una situación que desafía el sentido común y da como resultado una situación imposible. Aquí tienes 13 paradojas.

## 18 programas gratis para capturar pantalla en vídeo



Más de 18 programas gratuitos para crear screencasts: capturar o grabar en vídeo lo que hacemos en la pantalla de nuestro escritorio.

## ¿Qué significa G, E, 3G, H/3G+, H+, 4G?



¿Que son esos iconos y letras G, E, 3G, H, 3G+, H+, 4G que aparecen en nuestro smartphone? ¿A qué velocidad puedo descargar con cada uno?

## Internet más rápido (o cómo mejorar tu conexión)



10 consejos y trucos sobre cómo conseguir que nuestra conexión a Internet funcione mejor y más rápida.



[Artículos](#) [Feed RSS](#) [Aviso legal](#) [Privacidad & Cookies](#) [Publicidad](#) [Contacto](#)  
[HTML5](#) [CSS3](#)



Recibe artículos recién publicados

8 monos escribieron **268,61** páginas con sus máquinas de escribir en **0,02** segundos.  
CMS programado y diseñado por José Román Hernández Martín. Alojado en DigitalOcean usando CloudFlare.