MEDAC
Instituto Oficial de Formación Profesional

DISEÑO DE INTERFACES WEB

TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Multimedia: Animaciones e integración de contenido interactivo

08

ÍNDICE

OBJETIVOS

Identificar los formatos de imagen, audio y vídeo a utilizar.

Utilizar herramientas para el tratamiento de las imágenes.

Utilizar herramientas gráficas para el desarrollo de contenido multimedia interactivo.

Analizar el código necesario relacionado con la inclusión de contenido multimedia.

Añadir interactividad a elementos de un documento Web.

Verificar el funcionamiento de elementos multimedia e interactivos.

Realizar animaciones a partir de imágenes fijas.

/ 1. Introducción y contextualización práctica

Las animaciones, a diferencia de los elementos vistos en el tema anterior, resultan de la transición entre elementos a partir de un conjunto de propiedades definidas en CSS. Gracias a la inclusión de este lenguaje es posible crear animaciones de una manera más rápida y sencilla a un sitio web, aportando al mismo, un mayor dinamismo. Antes de la implementación con CSS era necesario crear script con JavaScript.

En este tema se verán las diferentes implementaciones posibles para la creación de animaciones: animation, transition o transfomation, entre otras.

Además, utilizando la librería JQuery en JavaScript es posible incorporar también cierto comportamiento interactivo a múltiples elementos que de otra forma serían estáticos.

La correcta selección de las animaciones y los comportamientos interactivos deben enriquecer el diseño del sitio web sin saturarlo de elementos inservibles.

Fig. 1. Ejemplo imagen pantalla código animaciones

Audio Intro. "Animaciones en un sitio web. ¿Son clave o un extra?"
https://bit.ly/3joBkhm

/ 2. 2. Características de las animaciones: animation

La creación de animaciones requiere de varios parámetros de diseño. De forma general se utiliza la propiedad animation seguida de alguno de los valores que se especifican en la siguiente tabla. Estas propiedades quedan incluidas dentro de un selector de la siguiente forma:

```
nombreSelector {
    animation-delay: ...;
    animation-name: nombreKeyFrame;
}
@keyframes nombreKeyFrame{
            ...
}
```

Código 1. Sintaxis básica inserción de animaciones

Por lo tanto, el funcionamiento de las animaciones queda definido por dos características esenciales: ***animation*** **y** ***keyframes.***

En primer lugar, se define el tiempo y el ritmo para cada animación, para ello se utiliza el elemento de la propiedad de animación correspondiente, el nombre de éstos queda formado con la palabra *animation* seguida de la propiedad que produce la animación. Por lo tanto, podemos distinguir entre los siguientes casos:

PARÁMETROS	DESCRIPCIÓN
animation-name	Indica el nombre de la regla @keyframes que implementa las propiedades de los fotogramas de la animación.
animation-delay	Indica el tiempo de retardo entre la carga de la animación y el comienzo de la reproducción de ésta.
animation-direction	Determina si cuando una animación termina vuelva reproducirse desde el inicio o se reproduce del final al principio.
animation-duration	Tiempo total que dura la animación.
animation-play-state	Permite pausar/reanudar la animación.
animation-timing-function	Determina el ritmo de la animación.
animation-fill-mode	Determina el valor que toman las propiedades de la animación después de la reproducción de ésta.
animation-iteration-count	Determina el número de veces que se repite el ciclo de la animación. Si se desea que sea de forma infinita se utiliza la palabra *infinite.*

Tabla 1. Opciones del elemento <animation>

/ 3. Características de las animaciones: keyframes

La propiedad *animation* permite definir en los selectores cómo se va a producir la transición entre los diferentes fotogramas que van a formar parte de la animación (el ritmo de transición, el número de ejecuciones, la duración...), es decir, permite definir la apariencia de la animación.

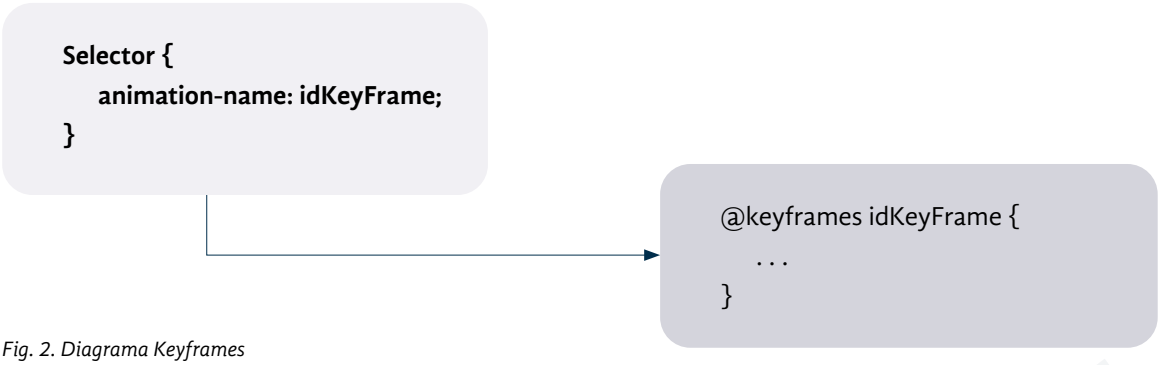

Fig. 2. Diagrama Keyframes

Hasta ahora no se ha definido el aspecto que tendrán cada uno de los "fotogramas" que componen la animación, esto se lleva a cabo utilizando **@keyframes.**

Esta propiedad determina cómo se va a mostrar un elemento tanto en el inicio como en el final de la animación, así como en diferentes intervalos de tiempo.

La apariencia en el inicio y en el final se indica tras la etiqueta ***from*** y ***to*** respectivamente, equivalentes al 0% y 100%. Los instantes intermedios se definen utilizando tantos **porcentajes** como se necesiten. Es posible definir tantos estados para la animación como se requieran.

Finalmente, es importante indicar que siempre se va a utilizar la etiqueta **@keyframes** para indicar que se está modelando un elemento de este tipo, seguido de un **nombre identificativo** que será el utilizado en la propiedad ***animation-name.***

La sintaxis queda definida de la siguiente forma.

```
@keyframes nombreKeyFrame{
    from {
        //aspecto inicial
    }
    % {
        //aspectos intermedios
    }
    to {
        //aspecto final
    }
}
```

Código 2. Sintaxis básica keyframes

/ 4. Animaciones de texto e imágenes

Algunos de los estilos de animación más utilizados son las que permiten que los elementos sobre los que añaden el efecto aparezcan en la pantalla "con movimiento".

Por ejemplo, para conseguir que un párrafo de texto aparezca por un extremo de la pantalla y termine en el opuesto se utilizará el siguiente código que se describe paso a paso:

1. En primer lugar, se utiliza el selector que define el elemento párrafo creado en HTML, **<p>**. En este caso se indica la duración total de la animación (5s) y, además, se va a ejecutar de forma cíclica (infinite).

```
p {
   animation-duration: 5s;
   animation-iteration-count: infinite;
   animation-name: animaTexto;
}
```

2. A continuación, se crea el elemento @keyframe utilizando como nombre identificador el mismo que se utiliza en *animation-name.*

```
@keyframes animaTexto{ ... }
```

3. Ahora se definen todos los valores de intervalo. En este caso se han definido cuatro intervalos: el inicio y el fin, y dos paradas más en el 30% y el 70% del tiempo de ejecución. Para probar este ejercicio basta con crear un fichero HTML con un elemento tipo párrafo y definir el siguiente estilo en el fichero CSS o con la etiqueta *<style>*.

```
@keyframes animaTexto{
  from {
     margin-right: 90%;
     width: 100%
  }
30% {
  font-size: 300%;
  margin-right: 25%;
  width: 150%;
}
70% {
  font-size: 600%;
     margin-right: 50%;
  width: 120%;
}
  to {
     margin-right: 0%;
     width: 100%;
  }
}
```

Código 3. Ejemplo keyframes

Audio 1. "Macromedia Flash"
https://bit.ly/2HxXTDi

/ 5. Caso práctico 1: "Creación de una animación"

Planteamiento: Entre las funciones que proporciona el atributo de animación, se implementa la posibilidad de que la animación se repita un número determinado de veces.

Escribe el código necesario para que la animación cumpla las siguientes especificaciones:

- La animación se repita 10 veces.
- Además, haz que, en lugar de volver al inicio en cada iteración, vuelva del final al principio.
- La animación dure 15 segundos

Nudo: Para realizar este ejercicio resultan claves todas las opciones posibles para la propiedad ***animation-direction*** (recordemos que esta propiedad determina si cuando una animación termina, vuelve a reproducirse desde el inicio o se reproduce del final al principio).

- ***normal.*** Cada vez que termina un ciclo, la animación se reinicia al estado inicial y comienza desde el principio. Este es el comportamiento por defecto.
- ***alternate.*** La animación, al terminar un ciclo, invierte su dirección. Es decir, los pasos de la animación se ejecutan al revés. Además, las funciones de tiempo también se invierten; por ejemplo, una animación *ease-in* (que veremos a continuación) se convierte en una animación con *ease-out* cuando se reproduce al revés. El contador que determina si la iteración es par o impar comienza en uno.
- ***reverse.*** Cada ciclo de la animación se reproduce al revés. Cada vez que comienza un ciclo de animación, ésta se posiciona en el estado final y comienza desde ahí.
- ***alternate-reverse.*** Es similar a *alternate* pero la animación se reproduce al revés. Es decir, la animación se posiciona en el estado final, comienza a reproducirse al revés y, cuando llega al inicio vuelve a reproducirse de forma normal hasta llegar al final de la secuencia. Esto vuelve otra vez a repetirse. El contador que determina si la iteración es par o impar comienza en uno.

Desenlace:

```
h1 {
    animation-duration: 15s;
    animation-direction reverse:
    animation-iteration-count:10;
}
```

Código 4. Código Caso práctico 1

/ 6. Transiciones

En primer lugar, uno de los elementos más utilizados para la creación de animaciones son las transiciones. Una transición supone un **cambio gradual entre elementos que será producida por una determinada acción,** por ejemplo, al pasar el puntero del ratón sobre una imagen, ésta cambiará progresivamente de color.

- El elemento ***transition*** permite indicar la propiedad que va a ver alterado su valor, así como la duración de este efecto. La **sintaxis básica de uso** desde un fichero CSS:

```
transition <propiedad> <duración> <tipo de acción> <retardo>
```

Código 5. Sintaxis básica inserción transición

- El atributo propiedad indica cuál va a ser la propiedad que va a modificar su estado, por ejemplo, el color de fondo *(background)*. La **duración** se utiliza para definir cuanto tiempo dura la transición ese define en segundos.

- El **retardo** se utiliza para indicar los intervalos de tiempo en los que se produce el cambio de estado en la propiedad, se define en segundos.

- Finalmente, el atributo **tipo de acción** permite definir el modo exacto de transición que se produce entre los estados del elemento. Los cambios de estado quedan definidos por los siguientes valores para esta propiedad:

VALOR	DEFINICIÓN
ease	La velocidad se incrementa al principio de la transición y se reduce al final. Si no se define ningún tipo de acción se toma ésta por defecto.
ease-in	La velocidad se incrementa desde el principio hasta el final de la transición.
ease-out	La velocidad se decrementa desde el inicio al final, es decir, comienza rápido y luego va disminuyendo la velocidad de la transición.
ease-in-out	La velocidad inicial y final es más lenta que la del tiempo central de la transición.
linear	Siempre tiene la misma velocidad.

Tabla 2. Valor cambios de estado.

Las propiedades que permiten modificar su valor utilizando las transiciones son múltiples, algunas de las más comunes son: *backgroud-color, color, width, height, opacity, margen, border, top, botón, left o right* entre otras.

Video 1. "Animaciones en la web"
https://bit.ly/3ouGPyH

/ 7. Transformaciones

Las transformaciones también son **utilizadas para crear sobre los elementos, ciertos efectos que simulan una animación.** Mientras que las transiciones modifican propiedades concretas, las transformaciones van a **rotar, desplazar y escalar** aquel elemento sobre el que son implementadas.

La propiedad ***transform*** define cómo posición origen para realizar la transformación, el eje central del elemento. Si lo que se busca es que este origen sea el lado superior izquierdo del objeto se utilizará la propiedad ***transform-origin.***

La sintaxis de uso desde un fichero CSS o desde entre las etiquetas <style> se muestra a continuación:

```
transform <scale> <translate> <rotate> <skew>n <matrix>
```

Código 6. Sintaxis básica transformación

- ***scale.*** Cambia las dimensiones del elemento seleccionado.
- ***skew.*** Modifica y distorsiona el aspecto de un elemento.
- ***translate.*** Desplaza el objeto. Permite seleccionar hacia donde se realiza la translación (translateX (valor en px), translateY (valor en px)).
- ***rotate.*** Gira el elemento los grados que se indican como parámetro (*rotateX* (valor en deg), *rotateY* (valor en deg)).

La propiedad *transform* se puede utilizar junto a la propiedad *transition,* esto resulta de utilidad cuando se pretende que la animación comience ante la ocurrencia de un determinado evento.

Por ejemplo, en el siguiente caso se han definido unas propiedades iniciales de animación y de aspecto para un bloque, ahora bien, cuando el puntero del ratón pasa por encima del elemento *(:hover)* este cambia y se produce la animación implementada.

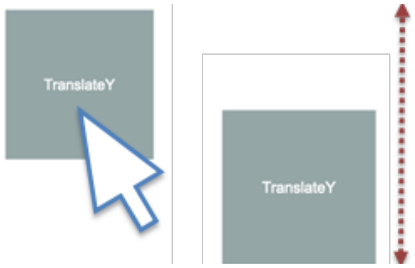

Fig. 3. Desplazamiento eje Y ejemplo

```
.nombreElemento{
    background-color: #95A5A6;
    transition: 2s linear;
}
 .nombreElemento:hover{
 transform: traslate(60px);
}
```

Código 7. Ejemplo transformación

/ 8. Steps

Tanto las animaciones como las transiciones y transformaciones vistas hasta ahora son algunas de las novedades que aparecieron con CSS3.

Las animaciones generadas utilizando las transformaciones, transiciones y animaciones descritas en los apartados anteriores, permiten diseñar una secuencia lineal, pero utilizando la función ***steps(),*** será posible controlar de forma completa el movimiento de las animaciones.

Esta función recibe por parámetro el número exacto de *frames* en los que va a quedar dividida la animación de la siguiente forma:

```
steps(número de frames);
```

Por ejemplo, si tenemos una animación creada con las transiciones que permitían modificar varias propiedades de los elementos, al utilizar la función *steps()* podemos definir el número de *frames* exactos en los que se producirá el cambio.

Veamos el funcionamiento de una forma práctica con el siguiente código. La sección identificada como "caja" cambiará de color en 63"pasos" o *"frames"* claramente diferenciados cuando el elemento es apuntado con el puntero del ratón *(:hover)*. Si no se utiliza esta función la transición sería lineal.

Ahora bien, desde CSS esto requiere de dos bloques, el primero donde se indique la apariencia inicial, y el segundo, en el que se implementa la apariencia final. La animación será el resultado de la transición entre ambos estados.

Será en el bloque correspondiente al estado inicial (section . caja{...}) donde se incorpora la función *steps(),* en concreto, sobre el elemento de aspecto que va a participar en la animación.

```
section .caja{
    background: #95A5A6;
    transition: background 2s steps(3);
}
section .caja:hover{
    background: #2980B9;
}
```

Código 8. Ejemplo steps()

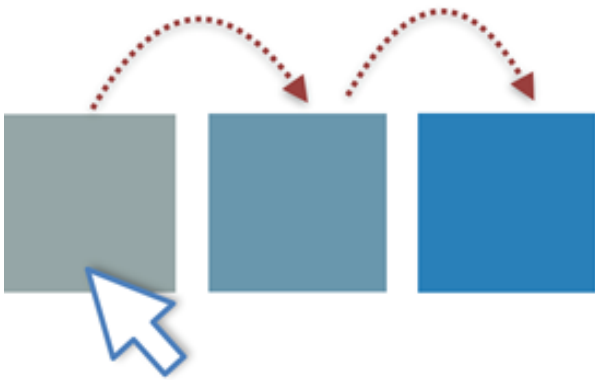

Fig. 4. Cambio de color steps() ejemplo

/ 9. Interactividad con JQuery: animate

Cuando trabajamos con JQuery para la inclusión de comportamiento interactivo son muchos los recursos que se pueden utilizar para conseguir este objetivo.

Existen funciones de carácter más particular que permiten la inclusión de efectos y se verán en el apartado siguiente.

La puesta en funcionamiento de esta librería es sencilla, basta con descargarla desde el sitio web (enlace) y añadir las siguientes etiquetas en el fichero HTML.

```
<script type="text/javascript"

src="jquery.js"></script>
```

Código 9. Sintaxis inclusión jquery.js

También es posible realizar la carga de la librería de forma directa apuntando a alguno de los repositorios que permiten el acceso a esto, la desventaja de esta forma es que solo funciona si hay conexión a Internet.

A continuación, para introducir la interactividad a un elemento basta con seleccionarle, para ello se utiliza la función **$(),** la cual nos permite poner el foco en prácticamente cualquier elemento del **DOM.** Algunos de los más utilizados son:

- La selección de elementos con **identificador:** $("#ID").
- La selección de elementos con **etiqueta:** $("p").
- La selección de elementos atendiendo a su **clase:** $(".nombreClase").
- La selección de **atributos** de un elemento: $("a[rel]").

En jQuery encontramos una función que permite la implementación de animaciones de forma directa, **animate(),** para ello recibe como parámetro obligatorio las propiedades que van a alterar su valor, así como otros de carácter opcional tales como la duración de la animación. Al tratarse de una función también es imprescindible indicar el elemento sobre el que se va a actuar.

```
$(elemento).animate({
  propiedades
}, [duración, easing, callback] );
```

Código 10. Selección elemento animación

Algunos de los parámetros optativos son:

- **Duración.** Se define en milisegundos. También se puede utilizar 'slow', 'normal' o 'fast'.
- ***Easing.*** Define la velocidad de la animación. 'linear' mantiene una velocidad constante.
- ***Callback.*** Define la función que se ejecuta al finalizar la animación.

Video 2. "JQuery para la creación de animaciones"
https://bit.ly/3kuJINR

/ 10. Efectos

Los elementos interactivos también permiten la inclusión de animaciones a través de los llamados efectos. Estos se implementan a través de la librería jQuery. Esta librería incorpora varias funciones que permiten añadir efectos a casi cualquier elemento definido en el DOM.

FUNCIÓN	DESCRIPCIÓN
.show	Esta función muestra el elemento sobre el que está definido.
.hide	Esta función oculta el elemento sobre el que está definido.
.slideDown	Esta función muestra el elemento sobre el que está definido. Esta entrada se realiza con un desplazamiento vertical.
.slideUp	Esta función oculta el elemento sobre el que está definido. Esta salida se realiza con un desplazamiento vertical.
.fadeIn	Esta función modifica la opacidad de un elemento al 100%.
.fadeOut	Esta función modifica la opacidad de un elemento al 0%.

Tabla 3. Efectos

La sintaxis de uso se muestra a continuación, donde en primer lugar se indica el elemento sobre el que se va a implementar la acción seguido de la función que define el efecto.

```
$('.nombreElemento').funciónEfecto( duración, callback);
```

Código 11. Sintaxis asignación efecto

Las funciones recogidas en la tabla anterior van a permitir recibir como **parámetros opcionales** la duración en milisegundos de la animación que determina la velocidad de esta. Por otro lado, de manera opcional permiten definir si una función ha de ser ejecutada tras la finalización de la animación.

Por ejemplo, para conseguir que un elemento se desvanezca por completo durante 5 segundos se utilizará el siguiente fragmento:

```
$('.nombreElemento').fadeOut(500);
```

Código 12. Ejemplo sintaxis asignación efecto

Si, además, se necesita que al concluir el desvanecimiento se llame a una nueva función el código quedaría de la siguiente forma:

```
$('.nombreElemento').fadeOut(500,
function(){
    //acciones
});
```

Código 13. Ejemplo 2 sintaxis asignación efecto

/ 11. Caso práctico 2: “Creación de una transición”

Planteamiento: Diseña los códigos HTML y CSS necesarios para crear dos cuadrados cuyo lado sea de 100 px, de color verde. Cuando el puntero del ratón pase por encima de cualquiera de los cuadrados, éstos se convertirán en sendos rectángulos cuya altura será la misma que la del cuadrado, y el ancho, el triple. Además, el color cambiará a rojo.

Nudo: En primer lugar, desde el fichero HTML se modelan las clases que implementan cada uno de los cuadrados, es decir, desde HTML se crean los bloques y la zona de la pantalla que luego serán caracterizadas como cuadrados de color utilizando las hojas de estilo.

Fichero HTML:

```
<html>
 <head>
   <meta              http-equiv=”Content-Type”
   content=”text/html; charset=UTF-8” />
 </head>
 <body>
   <div CLASS=”cuadradro1”>
        <h1>1</h1>
   </div>
   <div CLASS=”cuadrado2”>
        <h1>2</h1>
 </div>
</body>
</html>
```

Código 14. Código HTML Caso práctico 2

Desenlace: El fichero CSS en el que se incluyen las propiedades para la creación de la animación se muestra a continuación:

Fichero CSS:

```
.cuadrado1 {
    width: 100px;
    height: 100px;
    background-color: green;
    transition: width 2s, height 2s, margin 2s;
    margin: 50px auto 0;
}
.cuadrado1:hover{
    width: 200px;
    margin: 0 auto;
    background-color: red;
}
.cuadrado2 {
    width: 100px;
    height: 100px;
    background-color: green;
    transition: width 2s, height 2s, margin 2s;
    margin: 50px auto 0;
}
.cuadrado2:hover{
    width: 200px;
    margin: 0 auto;
    background-color: red;
}
```

Código 15. Código HTML Caso práctico 2

/ 12. Resumen y resolución del caso práctico de la unidad

Como se ha visto en este tema, las animaciones y los comportamientos interactivos enriquecen el diseño del sitio web. La creación de animaciones se puede llevar a cabo utilizando diferentes propiedades y técnicas. En primer lugar, ***animation*** y ***keyframes,*** gracias a estas propiedades es posible definir el tiempo y el ritmo para cada animación, para ello se utiliza el elemento de la propiedad de *animation* correspondiente, el nombre de estos queda formado con la palabra *animation* seguido de la propiedad sobre la que se produce la animación.

Por otro lado, se han analizado las transiciones, estas suponen un cambio gradual entre elementos que se producirá por una determinada acción, por ejemplo, al pasar el puntero del ratón sobre una imagen esta cambiará progresivamente de color.

Las transformaciones también son utilizadas para crear sobre los elementos ciertos efectos que simulan una animación. Mientras que las transiciones modifican propiedades concretas, las transformaciones van a rotar, desplazar y escalar aquel elemento sobre el que son implementada.

Gracias a la librería JQuery de Javascript también es posible la inclusión de comportamiento interactivo, por ejemplo, se pueden crear script que permitan que un bloque sea ocultado o mostrado en función de una determinada interacción con el usuario.

Resolución del caso práctico inicial

Como se ha visto a lo largo del tema, el uso de elementos animados resultará de utilidad para el diseño de atractivas interfaces web en la actualidad.

Existen muchos tipos de tecnologías que pueden ser utilizadas para implementar este tipo de elementos, como son HTML5, CSS3 o JavaScript en combinación Jquery.

En respuesta a la pregunta sobre si las animaciones son un elemento complementario o si pueden llegar a aportar incluso más información que el propio texto, podemos concluir tras el análisis exhaustivo del tema y la puesta en práctica de los elementos de programación, que las animaciones son un complemento que se puede incorporar en ciertas partes del sitio web, para dar énfasis, indicar el estado de carga de una determinada acción, o poner el foco en un evento, entre otras muchas opciones, por lo tanto, éstas pueden considerarse más un elemento complementario.

Fig. 5. Imagen tecnologías en animación

/ 13. Bibliografía

García-Miguel, D. (2019). *Diseño de Interfaces Web* (1.a ed.). Madrid, España: Síntesis.

Gauchat, J. (2017). *El gran libro de HTML5, CSS3 y JavaScript* (3.a ed.). Barcelona, España: Marcombo.