

Tema 7 – Contenidos web interactivos

Diseño de Interfaces Web

2 de marzo de 2023

CIFP Carlos III - Cartagena

Santiago San Pablo

2º curso DAW

Contenido

# Índice de ilustraciones.

# Índice de tablas.

Resumen tema 7.

# 1.- Introducción.

En esta unidad hablaremos de la inclusión de elementos interactivos o en movimiento dentro de una página web. El empleo de estos elementos pretende animar al usuario a participar e interactuar con la página web. El uso de elementos en movimiento capta la atención del usuario y le anima a interactuar con la página web.

**El uso de elementos interactivos debe estar centrado en el usuario y tener una utilidad**. La inclusión de demasiados elementos interactivos dentro de una página web puede ser contraproducente ya que puede molestar y distraer al usuario. Ten en cuenta que en la mayoría de los casos los elementos interactivos tienen que tener un inicio y un fin (no es recomendable utilizar recursos que se estén moviendo por la pantalla de manera indefinida).

# 2.- Orígenes de los contenidos interactivos.

Los objetos interactivos y en movimiento eran muy populares al inicio de las páginas web. Al principio, las páginas web eran estáticas, y para animarlos, se utilizaba mucho las imágenes GIF.

* **El formato GIF permite crear imágenes con movimiento** como fondo o simplemente como elemento decorativo.

Posteriormente, se popularizó el uso de JavaScript, mediante el cual, se pueden hacer pequeñas acciones interactivas.

* **Problema de JavaScript**: en sus primeros años, no todos los navegadores soportaban este lenguaje, y **algunos navegadores web** como Internet Explorer **utilizaban su propia versión de JavaScript**, por lo que se debía programar dependiendo del navegador que se fuera a utilizar.

A lo largo de esta documentación haremos referencia a documentos en código HTML. Con el objetivo de simplificar los contenidos y facilitar la realización de prácticas estos ficheros no se han insertado en los contenidos y se adjuntan en un fichero aparte en formato zip

# 3.- CSS animation.

Esta herramienta permite crer animaciones para la web usando solamente HTML y CSS. De este modo, no necesitamos JavaScript o JQuery. Estas animaciones pueden ser realmente complejas, como por ejemplo esta:

[Alex the CSS Husky (codepen.io)](https://codepen.io/davidkpiano/pen/wMqXea)

En **CSS Animation** temenos disponibles un gran nº de animaciones que se pueden utilizar de forma fácil y sencilla.

* Las propiedades de **CSS Animation** las podemos dividir en los siguientes grupos:
  + **CSS Transforms**: son propiedades CSS en las que se aplican transformaciones de **color, forma, etc.**, a un objeto.
  + **CSS Transitions**: indican cómo debe realizarse la transición entre las propiedades visuales de un objeto.
    - Se indica el tiempo de duración, delay, modo de transición, etc.
  + **Keyframes**: nos permiten realizar **animaciones complejas que requieran distintas transiciones y transformaciones**.
    - En los keyframes **indicamos el estado en el que tiene que estar el objeto y la transición** que se debe hacer **entre cada uno de ellos**.

**Aviso**: Actualmente el estándar CSS animation está soportado por todos los navegadores modernos. Podemos tener problemas con algunos navegadores antiguos. Para verificar si tu navegador soporta CSS animation, visita la siguiente página:

["css animation" | Can I use... Support tables for HTML5, CSS3, etc](https://caniuse.com/?search=css%20animation)

## 3.1.- CSS transform.

Para empezar a trabajar con estr grupo de propiedades, vamos a crear el siguiente HTML de ejemplo:

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Ejemplo de CSS transform</title>

    <link rel="stylesheet" href="custom.css" />

</head>

<body>

    <section class="profile"><button>ESTO ES UN BOTÓN</button></section>

</body>

</html>

Además, crearemos una hoja de estilo llamada **custom.css** donde escribiremos las propiedades CSS:

body {

    background-color: whitesmoke;

}

.profile {

    width: 100%;

    height: 100vh;

    display: flex;

    justify-content: center;

    align-items: center;

}

button {

    padding: 1.5rem 2.5rem;

    border: none;

    border-radius: 2rem;

    background-color: #563d7c;

    color: white;

    font-weight: bolder;

    box-shadow: 0 10px #999;

}

button:active {

    box-shadow: 0 4px #666;

    transform: translateY(6px);

}

Cuando pulsamos en el botón se aplica la clase button:active y por lo tanto se mueve el botón 6 pixeles hacia abajo y se cambia el color y tamaño de la sombra

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**Existe una gran cantidad de valores que se puede dar a esta propiedad**:

* [CSS transform property (w3schools.com)](https://www.w3schools.com/cssref/css3_pr_transform.php)
* [CSS 2D Transforms (w3schools.com)](https://www.w3schools.com/css/css3_2dtransforms.asp)
* [transform - CSS | MDN (mozilla.org)](https://developer.mozilla.org/es/docs/Web/CSS/transform)

**Los valores más utilizados para la propiedad transform son los siguientes**:

* **Translate**:
  + **transform:translate**(50px,100px): mueve el objeto 50px a la derecha y 100px hacia abajo.
  + **transform:translateX**(-10px): mueve el objeto en el eje X hacia la IZQUIERDA.
  + **transform:translateY**(-20px): mueve el objeto hacia ARRIBA 20px.
* **Scale**:
  + **transform:scale(2,3)**: agranda el tamaño del objeto 2 veces en el eje X y 3 veces en el eje Y.
  + **transform:scaleX(0,5)**: hace el ancho del objeto la mitad más pequeño.
  + **transform:scaleY(0.66)**: hace el alto de la imagen un tercio más pequeña.
* **Rotate**:
  + **transform:rotate(45deg)**: rota el objeto 45 grados.
  + **transform:rotate(180deg)**: rota el objeto en el eje X por lo tanto, pone el objeto al revés.
  + **transform:rotate(180deg)**: rota el objeto en el eje Y, por lo que pone el objeto invertido como si se viera en un espejo.

Si queremos **aplicar varias transformaciones al mismo tiempo** simplemente debemos de escribirlas una detrás de la otras. Por ejemplo:

Transform: rotate (45deg) translateX (-100px)

**Consideraciones**:

* Si aplicamos una rotación a un objeto, debemos indicar la palabra “deg” para indicar que son grados.
* Los grados pueden ser positivos (en el sentido de las agujas del reloj) o negativos (al contrario).
* La mayoría de los objetos tienen 2 dimensiones por lo que si los rotamos en los ejes X o Y 90 grados estos desaparecen de la pantalla aunque siguen ahí

**Atención**: es muy importante especificar las unidades de trabajo (px, %, deg, etc.), ya que si no, es muy probable que la transformación no se aplique.

## 3.2.- CSS transitions.

Hasta ahora hemos modificado el aspecto visual de un objeto (cambios estáticos de un objeto HTML). **Para hacer que un objeto cambie de color o forma de manera progresiva, necesitamos utilizar CSS transitions**.

Para la aplicación de las transiciones en los siguientes ejemplos se utilizará el selector **:hover** de CSS. **De este modo cuando nos posicionemos sobre el elemento se empezará a aplicar la transición**.

Las **opciones** que debe tener una **transición** son:

* **transition-property**: Indica el nombre de la propiedad CSS a la que se le aplicará el efecto.
* **transition-duration**: indica el tiempo en segundos que tardará en efectuarse el efecto.
* **transition-timing-function**: indica la velocidad de la curva con la que se realizará la transición.
* **transition-delay**: indica el tiempo en segundos que tardará en empezar el efecto.

Para ver el funcionamiento de las diferentes funciones de curva, puedes acceder a la siguiente página web:

[CSS transitions | CSS transitions, CSS transforms and CSS animation (bradshawenterprises.com)](http://css3.bradshawenterprises.com/transitions/)

Los siguientes trozos de código determinan como aplicar estas propiedades a un documento HTML:

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <title>Ejemplo de CSS transform</title>

    <link href="custom.css" rel="stylesheet" type="text/css">

</head>

<body>

    <div class="warpper">

        <section id="opt01">

            <h2>Cambio de color</h3>

            <div class="cuadrado"> </div>

        </section>

    </div>

</body>

</html>

.warpper {

    width: 100%;

    padding: 20px;

}

.cuadrado {

    width: 150px;

    height: 150px;

    margin: 1rem 0 2rem 1rem;

    background: blue;

}

#opt01 .cuadrado {

    transition-property: background;

    transition-duration: 2s;

}

#opt01 .cuadrado:hover {

    background: red;

}

El ejemplo anterior establece la clase “cuadrado” que nos permite visualizar un cuadrado de 150px de tamaño. Además se establece como propiedad a modificar el color de fondo “**background**” y se especifica que el cambio de color debe tardar 2 segundos. Este cambio solo se realizará cuando coloquemos el ratón encima del cuadrado.

**¿Cómo podemos especificar esto de una forma abreviada**?

transition: [propiedad] [duración] [delay] [función\_de\_curva]

De este modo, con la siguiente línea, tendríamos los mismos resultados que en el ejemplo anterior:

Texto

Descripción generada automáticamente

**Conocer más sobre transiciones**: [CSS Transitions (w3schools.com)](https://www.w3schools.com/css/css3_transitions.asp)

## 3.3.- Keyframes.

Mediante los Keyframes, podremos **crear animaciones complejas mezclando las transformaciones y transiciones** que hemos visto hasta ahora. La estructura de los keyframes es la siguiente:

@keyframe [nombre] {

    from {

        [posición\_inicial]

    }

    to {

        [posición\_final]

    }

}

Donde:

* [nombre] es el nombre que le damos al keyframe para después referenciarlo en el código CSS.
* [posición\_inicial]: es la posición inicial del objeto.
* [posición­\_final]: es la posición final del objeto.

**Ejemplo**:

@keyframes corre {

    from {

        transform: translateX(0)

    }

    to {

        transform: translateX(1000px)

    }

}

Con este keyframe, estamos estableciendo que un objeto se mueva en el eje X 1000px hacia la derecha.

**Para aplicar esta animación deberemos indicar las siguientes propiedades**:

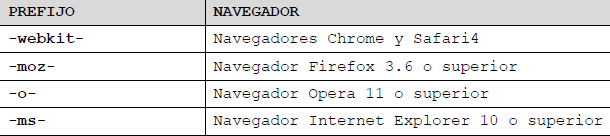
* **animation-name**: indica **qué keyframe** utilizar, y debe coincidir con el nombre dado a alguno de los keyframes establecidos. En el ejemplo anterior, es “corre”.
* **animation-duration**: indica cuánto **tiempo** durará la animación.
* **Animation**-function: curvas de Bezier.
* **animation-fill-node**: indica **qué pasará con el objeto cuando termine la animación**. Los **valores** que puede tener son:
  + **none**: valor por defecto. La animación aplicará ningún estilo ni antes ni después.
  + **Forwards**: el objeto se queda en la posición final del keyframe una vez finalizada la animación.
  + **Backwards**: el objeto está situado en la posición de inicio de la animación antes de que esta empiece.
  + **Both**: el objeto se posiciona en la posición de inicio antes de comenzar y se queda parado en la posición de finalización.
* **animation-delay**: retrasa el inicio de la animación el tiempo indicado.

**Sintaxis**: [animation - CSS | MDN (mozilla.org)](https://developer.mozilla.org/es/docs/Web/CSS/animation)

Animation: animation-name animation-duration animation-function animation-fill-node animation-delay

# 4.- Herramientas y utilidades.

Aunque CSS animations está ampliamente extendido entre los navegadores actuales, muchos navegadores tienen prefijos específicos. **Cada navegador ha creado un subconjunto de propiedades que soporta y que se diferencian unos de otros según el prefijo utilizado**.



Esto quiere decir que **para que sea totalmente compatible con navegadores más antiguos, deberemos aumentar el nº de líneas a escribir**.

-webkit-transition: background; /\* Chrome,Safari4+ \*/

-moz-transition: background; /\* FF3.6+ \*/

-o-transition: background; /\* Opera 11.10+ \*/

-ms-transition: background; /\* IE10+ \* transition: background; /\* propiedad estandar \*/

## 4.1.- Animate.css

Si queremos quitarnos el engorro de tener que escribir tanto código o simplemente queremos hacer animaciones senicllas, podemos utilizar la librería anímate.css, que se puede descargar en el siguiente enlace:

[Animate.css | A cross-browser library of CSS animations.](https://animate.style/)

En el siguiente vídeo podemos ver una explicación sencilla de cómo utilizar esta librería.

[Animar elementos html fácilmente con Animate.css - YouTube](https://www.youtube.com/watch?v=hlp-2wTFnCY)

## 4.2.- Animista.net

Esta página web **nos permite seleccionar la animación que queramos de forma sencilla**, viendo antes una pequeña presentación antes de aplicarla a cualquiera de los objetos de nuestra página HTML.

<https://animista.net/>

## 4.3.- Whirl.

Esta página contiene un montón de pequeñas animaciones que podemos utilizar cuando tengamos que cargar datos o esperar un tiempo hasta obtener el resultado final (PcComponentes cuando carga los artículos, YouTube cuando carga los vídeos, etc.).

[Whirl: CSS loading animations with minimal effort!](https://whirl.netlify.app/)

## 4.4.- Buttons generator.

Generador de multitud de animaciones para botones.

[Buttons Generator - Marko Denic - Web Developer](https://markodenic.com/tools/buttons-generator/)