C.F.G.S DE DESARROLLO DE APLICACIONES WEB

2º

DISEÑO DE INTERFACES WEB

Contenidos web interactivos

INDICE

Contenido

CONTENIDOS WEB INTERACTIVOS

1 INTRODUCCIÓN

En unidades anteriores hemos vistos como insertar recursos gráficos como imágenes, sonido, video o animaciones, estos complementos son importantes en una página web y que toda persona dedicada al diseño web debería conocer.

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.

Recuerda que el uso de elementos interactivos debe estar centrado en el usuario y debe 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 ORIGENES DE LOS CONTENDIOS INTERACTIVOS

Al contrario de lo que se pueda pensar en un principio los objetos interactivos y en movimiento eran muy populares al inicio de las páginas web. Al principio las páginas web eran documentos estáticos escritos en HTML y por lo tanto para animarlos se utilizaba mucho la inclusión de imágenes GIF.

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

http://toastytech.com/evil/index.html

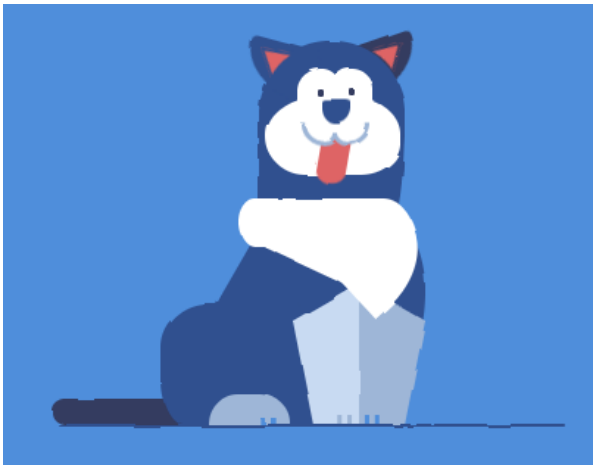
Posteriormente se popularizo el uso de JavaScript. Este lenguaje de programación se ejecuta en el navegador del usuario por lo que permitía realizar pequeñas acciones interactivas.

El principal problema de JavaScript en sus primeros años es que 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 de 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

CSS animation permite crear animaciones para la web usando solamente HTML y propiedades CSS. De este modo se elimina la necesidad de utilizar otros lenguajes como JavaScript o JQuery. Estas animaciones pueden ser realmente complejas como la realizada por David Khourshid. En la que anima a un Husky.



https://codepen.io/davidkpiano/pen/wMqXea

En **CSS Animation** tenemos disponibles un gran número 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. El objeto cambia sus propiedades "visuales" pero este no se mueve simplemente pasa de un estado a otro.
- **CSS TRANSITIONS:** Son propiedades CSS que indican cómo debe realizarse la transición entre las propiedades visuales de un objeto. En CSS transitions se indica el tiempo de duración, delay, modo de transición, etc.
- **KEYFRAMES:** Mediante Keyframes podemos 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.

ATENCIÓN

Actualmente el estándar CSS animation está soportado por todos los navegadores modernos. Podemos tener problemas con algunos navegadores antiguos ya que puede que no tengan todas las propiedades CSS animation soportadoas.

Para comprobar si tu navegador soporta CSS animation puedes visitar la siguiente página web.

https://caniuse.com/#search=css%20animation

3.1 CSS transform

Para empezar a trabajar con este grupo de propiedades vamos a crear el siguiente documento 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

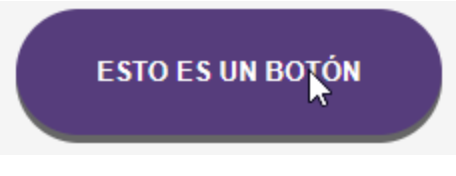
```
body {
    background-color: whitesmoke;

}
.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);
}
```

Esta hoja de estilo solo contiene una clase para centrar y alinear el botón que vamos a transformar. 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.



Existen una gran cantidad de valores que se puede dar a esta propiedad. Para ver todas las disponibles puedes acceder a la siguiente página web.

https://www.w3schools.com/cssref/css3_pr_transform.asp

https://www.w3schools.com/css/css3_2dtransforms.asp

https://developer.mozilla.org/es/docs/Web/CSS/transform

Los valores más utilizados para la propiedad transform son las siguientes:

VALOR	EJEMPLO	SIGNIFICADO
translate	`transform:translate(50px,100px)`	Mueve el objeto 50px a la derecha y 100px hacia abajo
translateX	`transform:translateX(-10px)`	Mueve el objeto en el eje X hacia la IZQUIERDA
translateY	`transform:translateY(-20px)`	Mueve el objeto hacia ARRIBA 20px
scale	`transform:scale(2,3)`	Agranda el tamaño de la objeto 2 veces en el eje X y 3 veces en el eje Y
scaleX	`transform:scaleX(0,5)`	Hace el ancho del objeto la mitad más pequeño
scaleY	`transform:scaleY(0,66)`	hace el alto de la imagen un tercio más pequeña
rotate	`transform:rotate(45deg)`	Rota el objeto 45 grados
rotateX	`transform:rotate(180deg)`	Rota el objeto en el eje X por lo tanto pone el objeto al revés.
rotateY	`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)
```

A la hora de aplicar transformaciones ten en cuenta las siguientes consideraciones:

- Los números positivos mueven los objetos hacia la derecha o hacia abajo. Los negativos a la izquierda o hacia arriba.
- Si aplicamos una escala de (1) a un objeto este no cambia de tamaño. Si el valor es mayor de 1 se hace más grande y si el valor es menor de 1 se hace más pequeño.
- Si aplicamos una rotación a un objeto debemos indicar SIEMPRE la palabra “deg” para indicar que son grados.
- Los grados pueden ser positivos (en sentido de las agujas del reloj) o negativos (al contrario de las agujas del reloj)
- 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

Un error muy común es no indicar las unidades de trabajo, px, %, deg, etc. Si no indicamos la unidad es muy probable que la transformación no se aplicuque.

3.2 CSS transitions

Hasta ahora hemos modificado el aspecto visual de un objeto dentro de HTML pero ese cambio es estático y no se produce ninguna animación entre el estado inicial y el nuevo aspecto. 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:

PROPIEDAD	SIGNIFICADO
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.

Puedes ver el funcionamiento de las diferentes funciones de curva accediendo a la siguiente página web.

```
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="estilo1.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.

Para reducir el número de líneas a rellenar podemos comprimir las especificaciones anteriores con la propiedad **transition** la cual tiene la siguiente estructura.

```
transition: [propiedad] [duración] [delay] [función_de_curva]
```

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

```
#opt01 .cuadrado {
      transition: background 2s;
}
#opt01 .cuadrado: hover {
      background: red;
}
```

Puedes ver un ejemplo completo de las transiciones en el fichero adjunto.

```
css-transitions.html
```

También puedes conocer más sobre transiciones en el siguiente enlace.

https://www.w3schools.com/css/css3_transitions.asp

Puedes ver un ejemplo un ejemplo más complejo del uso de transiciones en el siguiente fichero.

```
css-transitons-ej01.html
```

3.3 Keyframes

Mediante los Keyframes podemos 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.

Un ejemplo sería el siguiente

```
@keyframes  corre {
    from {
        transform: translateX(0)}
    to {
        transform: translateX(1000px)}
}
```

Con este keyframe estamos estableciendo que un objeto se mueva en el eje x 1000 pixeles hacia la derecha.

Para aplicar esta animación deberemos de indicar las siguientes propiedades.

PROPIEDAD	**SIGNIFICADO**
animation-name	Indica que keyframe utilizar y debe coincidir con el nombre dado a algunos de los keyframes establecidos. En el ejemplo anterior es "corre"

animation-duration	Indica cuánto tiempo durará la animación.
animation-fill-mode	Indica que pasará con el objeto cuando termine la animación. Los valores que puede tener son: - **none:** valor por defecto. La animación no 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.

Puedes ver un ejemplo de cómo aplicar keyframes en el siguiente archivo adjunto.

`css-keyframes.html`

Puedes ver un ejemplo de cómo aplicar estas y otras propiedades en el siguiente ejemplo.

`css-keyframes-ej02.html`

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.

Así los prefijos para cada navegador son los siguientes:

PREFIJO	NAVEGADOR
-webkit-	Navegadores Chrome y Safari4
-moz-	Navegador Firefox 3.6 o superior
-o-	Navegador Opera 11 o superior
-ms-	Navegador Internet Explorer 10 o superior

Esto quiere decir que para ser totalmente compatible con navegadores más antiguos deberemos aumentar el número de líneas a escribir. Por ejemplo si queremos realizar una transición del color de fondo de un objeto deberíamos de 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 sencillas podemos utilizar la librería anímate.css que se puede descargar en el siguiente enlace.

```
https://daneden.github.io/animate.css/
```

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

```
https://youtu.be/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

En esta página podemos seleccionar el efecto de animación que deseemos y después dar en el botón de generar código. De este modo nos generará el código que tenemos que incrustar en nuestro fichero CSS para que se reproduzca en la mayoría de los navegadores del mercado.

4.3 Whirl

Está página contiene un motón de pequeñas animaciones que podemos utilizar en los momentos en los que tengamos que cargar datos o esperar un tiempo hasta obtener el resultado final.

```
https://whirl.netlify.app/
```

4.4 Buttons generator

Generador de multitud de animaciones para botones.

```
https://markodenic.com/tools/buttons-generator/
```