Técnicas de Maquetación Web

Índice de contenidos del curso

Posicionar en capas

Cuando se crean capas o se colocan elementos (texto, imágenes, etc.) sin definir su posicionamiento, estas siguen el orden del flujo del documento y se colocan uno debajo del otro.

Pero puede que nosotros queramos determinar el tipo de posicionamiento que tendrán las capas y cualquier otro elemento. Para ello emplearemos el atributo “position”, que puede tener únicamente cuatro valores: “static”, “relative”, “absolute” y “fixed”.

Al colocar alguno de estos valores, conseguiremos posicionar las capas o elementos html de forma diferente.

Hasta ahora, hemos estado trabajando sin saberlo en lo que se denomina posicionamiento estático (*static*), donde todos los elementos aparecen con un orden natural según donde estén colocados en el HTML. Este es el modo por defecto en que un navegador renderiza una página.

Pero, como ya hemos adelantado, a la propiedad position se le pueden indicar los siguientes valores:

Propiedad	Valor	Descripción del ejemplo
position	static	Posicionamiento estático. Utiliza el orden natural de los elementos HTML.
	relative	Posicionamiento relativo. Los elementos se mueven ligeramente en base a su posición estática.
	absolute	Posicionamiento absoluto. Los elementos se colocan en base al contenedor padre.
	fixed	Posicionamiento fijo. Idem al absoluto, pero aunque hagamos scroll no se mueve.
	sticky	Posicionamiento «pegado». Similar al relativo, usado para pegar menús a la parte superior.

Si utilizamos un modo de posicionamiento diferente al estático (*absolute*, *fixed*, *sticky o relative*), podemos utilizar una serie de propiedades para modificar la posición de un elemento. Estas propiedades son las siguientes:

Propiedad	Valor	Descripción del ejemplo
top:	**auto | size**	Empuja el elemento una distancia desde la parte superior hacia el inferior.
bottom:	**auto | size**	Empuja el elemento una distancia desde la parte inferior hacia la superior.
left:	**auto | size**	Empuja el elemento una distancia desde la parte izquierda hacia la derecha.
right:	**auto | size**	Empuja el elemento una distancia desde la parte derecha hacia la izquierda.
z-index:	**auto | number**	Coloca un elemento en el eje de profundidad, más cerca o más lejos del usuario.

Posicionamiento relativo

Si utilizamos la palabra clave *relative* activaremos el modo de posicionamiento relativo, que es el más sencillo de todos. En este modo, los elementos se colocan exactamente igual que en el posicionamiento estático (*permanecen en la misma posición*), pero dependiendo del valor de las propiedades top, bottom, left o right podremos variar ligeramente la posición del elemento.

Posicionamiento absoluto

El valor *absolute* en el atributo position permite posicionar elementos de manera absoluta, esto es de manera definida por valores de los atributos top, left, bottom y right, que indican la distancia con respecto a un punto. Las capas o elementos con posicionamiento absoluto quedan aparte del flujo normal del HTML, quiere decir esto que no se afectan por el lugar donde aparezcan en el código HTML y tampoco afectan ellas a otros elementos que sí que formen parte del flujo normal del HTML.

Los valores top, left, bottom y right se expresan con unidades CSS y son una distancia con respecto al primer elemento contenedor que tenga un valor de position distinto de static. Si todos los contenedores donde esté la capa posicionada con position absolute (todos sus padres hasta llegar a BODY) son static, simplemente se posiciona con respecto al lado superior de la página, para el caso de top, el inferior para bottom, del lado izquierdo para left o el derecho, en el caso de utilizar right.

Posicionamiento fijo

El posicionamiento *fixed* es hermano del posicionamiento absoluto. Funciona exactamente igual, salvo que hace que el elemento se muestre en una posición fija dependiendo de la región visual del navegador. Es decir, aunque el usuario haga scroll y se desplace hacia abajo en la página web, el elemento seguirá en el mismo sitio posicionado.

Posicionamiento sticky

El posicionamiento *sticky* se suele utilizar cuando queremos que un elemento se posicione en un lugar específico de forma fija (*«sticky»*, *pegajoso*), como por ejemplo, cuando al hacer scroll llegamos a un elemento y queremos que ese elemento se quede fijo en la parte superior mientras continuamos haciendo scroll. Este comportamiento es muy habitual con los menús superiores de las páginas o las cabeceras de secciones.

Profundidad (niveles)

Es interesante conocer también la existencia de la propiedad *z-index*, que establece el nivel de profundidad en el que está un elemento sobre los demás. De esta forma, podemos hacer que un elemento se coloque encima o debajo de otro.

Su funcionamiento es muy sencillo, sólo hay que indicar un número que representará el nivel de profundidad del elemento. Los elementos con un número más alto estarán por encima de otros con un número más bajo, que permanecerán ocultos detrás de los primeros.

Propiedad Display

Cada elemento HTML tiene un tipo de representación concreto. Como norma general (*con excepciones*) los elementos que se utilizan dentro de un párrafo, son de tipo inline, mientras que los que se utilizan para agrupar, son de tipo block. La propiedad display de CSS permite modificar el comportamiento de un elemento HTML, cambiándolo al que le indiquemos, como por ejemplo inline o block.

Existe una amplia gama de tipos de representación de elementos HTML que podemos utilizar mediante la propiedad display.

Propiedad	Tipo de Caja	Característica
display	block	Se apila en vertical. Ocupa todo el ancho disponible de su etiqueta contenedora.
	inline	Se coloca en horizontal. Se adapta al ancho de su contenido. Ignora width o height.
	inline-block	Combinación de los dos anteriores. Se comporta como inline pero no ignora width o height.
	flex	Utiliza el modelo de cajas flexibles de CSS. Ideal para estructuras de 1 dimensión.
	inline-flex	Versión en línea (ocupa sólo su contenido) del modelo de cajas flexibles de CSS.
	grid	Utiliza cuadrículas o rejillas con el modelo de cajas Grid CSS.
	inline-grid	La versión en línea (ocupa sólo su contenido) del modelo de cajas grid css.
	list-items	Actúa como un ítem de una lista. Es el comportamiento de etiquetas como .
	table	Actúa como una tabla. Es el comportamiento de etiquetas como <table>.
	table-cell	Actúa como la celda de una tabla. Es el comportamiento de etiquetas como <th> o <td>.
	table-row	Actúa como la fila de una tabla. Es el comportamiento de etiquetas como <tr>.
	contents	Ignora la caja del elemento. Útil para mantener Grid/Flex aún teniendo un wrapper intermedio.
	none	Oculta el elemento visualmente, como si no existiera en el HTML.

Propiedad float

Con *float* podemos conseguir que un elemento «flote» a la izquierda o a la derecha de otro elemento.

Por otro lado, la propiedad clear se encarga de impedir elementos flotantes en la zona indicada, a la izquierda del elemento (*left*), a la derecha (*right*) o en ambos lados (*both*).

Propiedad	Valor	Descripción del ejemplo
float	none | left | right	Cambia el flujo para que el elemento flote a la izquierda o derecha.
clear	none | left | right | both	Impide que los elementos puedan flotar en la orientación indicada.

Transformaciones

En CSS podemos realizar transformaciones gracias a la propiedad *transform*. A esta propiedad se le aplican una serie de valores, los cuales serán aplicados a los elementos HTML seleccionados, haciendo que estos se transformen.

Las transformaciones pueden ser en 2D y en 3D. Veremos primeramente las transformaciones en 2D.

Tenemos 4 tipos de transformaciones:

Tipo	Descripción
posición / translación	Desplaza un elemento en el eje X (*izquierda, derecha*) y/o en el eje Y (*arriba, abajo*)
escala	Escala el elemento una determinada cantidad más grande o más pequeña. También se puede voltear.
rotación	Gira el elemento sobre su eje X o sobre su eje Y. También se puede girar sobre sí mismo.
deformación	Inclina el elemento sobre su eje X o sobre su eje Y.

Veremos que en este tema de las transformaciones podemos trabajar de dos maneras: o bien asignando valores de translación ("*translate*"), escala ("*scale*"), rotación ("*rotate*"), o deformación ("*skew*") a la propiedad transformar ("*transform*"). Hasta ahora se ha usado esta:

```
.class {
        transform:rotate(30deg) translate(50%,0) scale(1.2);
}
```

Los nuevos navegadores nos permiten reescribir nuestro CSS de tal manera que podemos utilizar esos valores que asignamos a la propiedad transformación de manera independiente, como propiedades CSS.

Si lo hiciéramos de esa manera, el código que acabamos de escribir podríamos reescribirlo de esta otra forma:

```
.class {
        translate: 50% 0;
        rotate: 30deg;
        scale: 1.2;
 }
```

asignando valores de translación (“*translate*”), escala (“*scale*”), rotación (“*rotate*”), o deformación (“*skew*”) a la propiedad transformar (“*transform*”). Hasta ahora se ha usado esta:

Diferencias entre ambas opciones:

Cuando usamos la propiedad ***transform*** y encadenamos varios valores, estos se aplicarán al elemento en el **orden en el que los hayamos puesto en nuestro CSS**.

Si aplicamos **translate, rotate y scale como propiedades**, y no como valores de la propiedad transform, estos si o si siempre seguirán un orden a la hora de ser aplicados. Este orden será:

1. **translate**
2. **rotate**
3. **scale**

Punto de origen

Cuando utilizamos la propiedad transform, adicionalmente podemos usar la propiedad *transform-origin*. Esta propiedad nos permite cambiar el punto de origen de una transformación. Los valores que asignaremos a esta propiedad corresponderán a los parámetros de la posición de origen de cada eje (*X e Y*), que podemos indicar, por ejemplo, con porcentajes, y que por defecto, está establecida a 50% 50%.

Propiedad	Valores	Descripción
transform-origin	posiciónX posiciónY	Cambia el punto de origen del elemento en una transformación.

Función de translación (translate como valor)

Las funciones de translación nos permiten mover un elemento de un lugar a otro. Para ello, las utilizaremos en el interior de la propiedad CSS transform y elegiremos una de las siguientes funciones de translación.

Si especificamos un valor positivo en el eje X (*horizontal*), lo moveremos hacia la derecha, y si especificamos un valor negativo, lo moveremos hacia la izquierda. Ocurre lo mismo con el eje Y (*vertical*), que con valores negativos lo movemos hacia arriba y con valores positivos lo movemos hacia abajo.

Funciones	Significado
translateX(x)	Traslada el elemento una distancia de x horizontalmente.
translateY(y)	Traslada el elemento una distancia de y verticalmente.
translate(x, y)	Propiedad de atajo de las dos anteriores.
translate(x)	Equivalente a translate(x, 0)

Propiedad translate

Esta propiedad está soportada por nuevos navegadores. En este caso la usamos como propiedad (es decir, translate: valor;) y , por lo tanto, no necesita para nada la propiedad *transform*.

En este caso, se le pueden asignar hasta 3 valores ya que nos permite desplazar los elementos en los 3 ejes: X Y Z

Propiedad	Valor	Descripción
translate	none	No aplica desplazamiento. Valor por defecto.
translate	valor	Desplaza un elemento el valor especificado en el eje X.
translate	valor valor	Desplaza un elemento una cierta cantidad en el eje X y eje Y.
translate	valor valor valor	Desplaza un elemento el eje X, eje Y y eje Z.

OJO!!

En el caso de indicar sólo un parámetro, los demás ejes serán 0px por defecto.

Función de rotación(rotate como valor)

La función de rotación nos permite girar un elemento una cierta cantidad respecto al eje involucrado. Disponemos de las siguientes funciones de rotación:

Funciones	Significado
rotateX(x)	Establece una rotación 2D en x sólo para el eje horizontal X.
rotateY(y)	Establece una rotación 2D en y sólo para el eje vertical Y.
rotateZ(z)	Establece una rotación 2D en z sobre si mismo.
rotate(z)	Alias a la anterior.

Propiedad rotate

Al igual que pasa con la propiedad *translate*, En nuevas versiones de los navegadores, ya se soporta la propiedad individual *rotate*, y no hace falta utilizarla dentro de la propiedad *transform*.

Propiedad	Valor	Descripción
rotate	none	No aplica rotación. Valor por defecto.
rotate	ángulo	Rota el elemento sobre si mismo. Equivalente a rotateZ().
rotate	eje ángulo	Rota el elemento sobre el eje (x, y o z) indicado.
rotate	nº nº nº ángulo	Indica un vector de rotaciones con el ángulo indicado.

Función de escalado(scale como valor)

Esta función realiza una transformación en la que aumentan o reducen el tamaño de un elemento. Para ello, la utilizaremos en el interior de la propiedad CSS *transform* y elegiremos una de las siguientes funciones de escalado.

Funciones	Significado
scaleX(fx)	Reescala el elemento un número de *fx* veces (sólo en horizontal).
scaleY(fy)	Reescala el elemento un número de *fy* veces (sólo en vertical).
scale(fx, fy)	Propiedad de atajo de las dos anteriores (escalado simétrico).
scale(fx)	Equivalente al anterior: scale(fx, fy).

Propiedad scale

Al igual que en casos anteriores, nuevas versiones de los navegadores, ya se soporta la propiedad individual *scale*, y no hace falta utilizarla dentro de la propiedad *transform*.

Propiedad	Valor	Descripción
scale	none	No aplica escalado. Valor por defecto.
scale	nº	Aplica el factor de escala simétrico a los dos ejes X/Y. Igual a scale: x x 1.
scale	nº nº	Aplica los factores de escala al eje X y eje Y. Igual a scale: x y 1
scale	nº nº nº	Aplica un factor de escala a cada eje. Igual a scale: x y z.

Nota: Podríamos utilizar porcentajes en vez de números.

CURIOSIDAD!!

Si usamos valores negativos, se creará un efecto espejo.

Función de deformación(skew como valor)

La función deformación establece un ángulo para torcer, tumbar o inclinar un elemento en 2D. Tenemos disponibles las siguientes:

Funciones	Significado
skewX(xdeg)	Establece un ángulo de *xdeg* para una deformación 2D respecto al eje X.
skewY(ydeg)	Establece un ángulo de *ydeg* para una deformación 2D respecto al eje Y.

En este caso, skew no funciona como propiedad, únicamente como función o valor de transform.

Transformaciones 3D

Estas funciones no son más que la incorporación del eje Z a las anteriores que ya habíamos visto, además de la adición de una función de atajo 3D para poder utilizarlas todas de una sola vez.

Recordemos que el eje X es el eje horizontal, el eje Y es el eje vertical y el eje Z es el eje de profundidad.

Para utilizar transformaciones 3D es necesario conocer algunas propiedades derivadas de transformaciones, como por ejemplo, las siguientes:

Propiedad	Valores	Descripción
transform-style	flat | preserve-3d	Modifica el tratamiento 3D de los elementos hijos.
transform-origin	X Y Z	Cambia el punto de origen del elemento en una transformación.

Si utilizamos la propiedad *transform-style:preserve-3d* todos los elementos hijos del elemento que tenga esa propiedad, se tratarán como elementos 3D.

Función de translación 3D

Las funciones de transformación que completan la colección de transformaciones 2D que vimos anteriormente son:

Función	Significado
translateZ(z)	Traslada el elemento una distancia de z en el eje de profundidad.
translate3d(x, y, z)	Función de atajo 3D, donde podemos aplicar un a x, y y z.

```
.element {
 width: 100%;  height: 100%;  background: red;
 transform: translateX(40px);    /* Sólo eje X */  transform: translateY(25px);    /* Solo eje Y */  transform: translateZ(55px);    /* Solo eje Z */
 transform: translate3d(40px, 25px, 55px);   /* Equivale a los tres anteriores */
}
```

Función de rotación 3D

Tenemos las funciones de rotación 3D. RotateZ() es una función que permite indicar la rotación sobre su propio eje, algo que podíamos hacer ya en el ámbito 2D.

Sin embargo, se añade la propiedad rotate3d(), que nos permite indicar la cantidad de grados que queremos realizar en cada eje, de una forma muy flexible:

Función	Significado
rotateZ(zdeg)	Aplica rotación 2D en *zdeg* grados sólo para el eje Z (profundidad).
rotate3d(x, y, z, deg)	Función de atajo, que permite aplicar una rotación a uno o varios ejes.

```
.element {
 transform: rotate3d(1, 0, 0, 50deg);  /* Equivale a rotateX(50deg) */
 transform: rotate3d(0, 1, 0, 25deg);  /* Equivale a rotateY(25deg) */
 transform: rotate3d(0, 0, 1, 5deg);   /* Equivale a rotateZ(5deg) */

 transform: rotate3d(1, 1, 0, 100deg); /* Equivale a rotateX(100deg) rotateY(100deg) */
}
```

Función de escalado 3D

De la misma forma que las anteriores que hemos visto, las funciones de escalado también tienen su versión 3D.

Para que el escalado de Z sea evidente, quizás sería conveniente mirar primero el tema de perspectivas.

Función	Significado
scale>(fz)	Reescala el elemento a un nuevo tamaño con factor *fz* de profundidad.
scale3d(fx, fy, fz)	Establece un escalado 3D, donde aplica los factores a cada eje.

```
.element {
 transform: scaleZ(2);        /* Reescala al doble de su tamaño el eje Z */
 transform: scale3d(1, 0.5, 2); /* Equivale a scaleX(1) scaleY(0.5) scaleZ(2) */
}
```

Perspectiva

Cuando trabajamos con 3D en CSS, en muchas ocasiones es necesario dotar a nuestro trabajo de perspectiva. Con la propiedad perspective de CSS podemos establecer un punto de fuga con una cierta distancia a un elemento contenedor para dotar de perspectiva a sus elementos hijos.

Propiedades	Formato	Significado
perspective	none | tamaño	Punto de fuga para los elementos hijos.
perspective-origin	X Y	Punto de origen de la perspectiva.
backface-visibility	visible | hidden	Oculta la cara posterior o subelemento que nosotros designemos de los que forman un elemento 3D.

Transiciones

Las transiciones se basan en un principio muy básico: conseguir un efecto suavizado entre un estado inicial y un estado final al realizar una acción.

Las propiedades CSS que podemos utilizar relacionadas con las transiciones son las siguientes:

Propiedades	Valor	Significado
transition-property	all | none | propiedad CSS	Propiedades CSS afectadas por la transición.
transition-duration	0 | tiempo	Tiempo de duración.
transition-timing-function	Funciones que veremos más adelante	Ritmo de la transición.
transition-delay	0 | tiempo	Tiempo de retardo inicial.

transition-property

Se utiliza para especificar la propiedad a la que afectará la transición. Podemos especificar la propiedad concreta (*width o color, por ejemplo*) o simplemente especificar *all* para que se aplique a todos los elementos con los que se encuentre. Por otro lado, *none* hace que no se aplique ninguna transición.

Propiedades		Valor
all		Aplica la transición a todas las propiedades css.
none		No aplica transición. El cambio se producirá de golpe (*brusco*).
propiedad CSS		Aplica la transición sólo a la propiedad css especificada.

transition-duration

Con esta propiedad especificamos la duración de la transición, desde el inicio de la transición, hasta su finalización. Se recomienda siempre comenzar con valores cortos, para que las transiciones sean rápidas y elegantes. Por defecto, las transiciones tienen una duración de 0s, por lo que si no cambiamos este valor, será lo mismo que no tener transición.

transition-delay

La propiedad transition-delay nos ofrece la posibilidad de retrasar el inicio de la transición un número de segundos determinado. Si se omite, la transición comienza inmediatamente.

Ej:

```
a {
 background: #ff0000;
 color: #ffffff;
 padding: 8px;
}

a:hover {
 background: #fff;
 color: #ffffff;
 padding: 25px;
 border: 1px solid #888;

 transition-property: all;
 transition-duration: 0.5s;
 transition-timing-function: linear;
}
```

=

```
a {
 background: #000000;
 color: #222;
 padding: 8px;
}

a:hover {
 background: #fff;
 color: #ffffff;
 padding: 25px;
 border: 1px solid #888;

 transition: all 0.5s linear;

}
```

Transiciones de entrada y salida

Muchas de las transiciones que aplicamos lo hacemos a elementos al colocar el ratón sobre un elemento.

Lo mismo que aplicamos una transición a un efecto cuando colocamos el ratón sobre un elemento, podemos hacer que, al quitar el ratón de ese elemento, este vuelva a su estado original con una pequeña transición también. Vemos un ejemplo

```
a {
 background: #ff0000;
 color: #ffffff;
 padding: 8px;
transition: background 2s linear;

}

a:hover {
 background: #000000;
 transition: background 2s linear;
}
```

Funciones de tiempo

Tanto cuando hablamos de transiciones como cuando hablamos de animaciones tenemos dos propiedades que se aplican exactamente igual y se encargan de definir el ritmo o transcurso de la animación o transición en cuestión.

Propiedades	Descripción
transition-timing-function	Ritmo de la transición
animation-timing-function	Ritmo de la animación

Ambas propiedades admiten cualquiera de los siguientes valores que vamos a ver a continuación:

Si no se indica ninguna función de tiempo concreta, CSS utilizará la función de tiempo ease.

En el caso de que ninguna de ellas nos ofrezca el ritmo que buscamos, podemos definirla nosotros mismos manualmente mediante cubic-bezier().

Valor	Inicio	Transcurso	Final	Equivalente en cubic-bezier
ease	Lento	Rápido	Lento	cubic-bezier(0.25, 0.1, 0.25, 1)
linear	Normal	Normal	Normal	cubic-bezier(0, 0, 1, 1)
ease-in	Lento	Normal	Normal	cubic-bezier(0.42, 0, 1, 1)
ease-out	Normal	Normal	Lento	cubic-bezier(0, 0, 0.58, 1)
ease-in-out	Lento	Normal	Lento	cubic-bezier(0.42, 0, 0.58, 1)
cubic-bezier(*A*, *B*, *C*, *D*)				Transición personalizada

Función cubic bezier

La función de tiempo *cubic-bezier()* es una función personalizada, donde podemos darle unos valores concretos dependiendo de la velocidad que queramos que tenga la transición.

En principio, el formato de la función es cubic-bezier(A, B, C, D), donde esos cuatro parámetros son números que indican lo siguiente:

Parámetro	Valor	Descripción
A	X_1	Eje X del primer punto que orienta la curva bezier.
B	Y_1	Eje Y del primer punto que orienta la curva bezier.
C	X_2	Eje X del segundo punto que orienta la curva bezier.
D	Y_2	Eje Y del segundo punto que orienta la curva bezier.

Animaciones

La diferencia entre las transiciones y las animaciones radica en que en este último caso no hace falta que el usuario interaccione con el elemento que tiene esta animación.

Las transiciones son una manera de suavizar un cambio de un estado inicial a un estado final. La idea de las animaciones CSS parten del mismo concepto, pero a diferencia de las transiciones, permiten añadir más estados aún. Así pues, con las animaciones podemos partir desde un estado inicial, a un estado posterior, a otro estado posterior, y así sucesivamente.

Para crear animaciones CSS es necesario realizar 2 pasos:

- Utilizar la propiedad *animation* (*o derivadas*) para indicar que elemento HTML vamos a animar.
- Definir mediante la regla *@keyframes* la animación en cuestión y sus estados (*fotogramas clave*).

Propiedades de animación

Para el comportamiento de una animación, necesitamos conocer las siguientes propiedades, que son una «ampliación» de las propiedades de las transiciones CSS:

Propiedades	Descripción	Valor
animation-name	Nombre de la animación a aplicar.	none | nombre
animation-duration	Duración de la animación.	0 | tiempo
animation-timing-function	Ritmo de la animación.	
animation-delay	Retardo en iniciar la animación.	0 | tiempo
animation-iteration-count	Número de veces que se repetirá.	1 | infinite | número
animation-direction	Dirección de la animación.	normal | reverse | alternate | alternate-reverse
animation-fill-mode	Como se «completa» la animación.	none | forwards | backwards | both
animation-play-state	Estado de la animación.	running | paused

Propiedad animation-name

Permite especificar el nombre de la animación (*definido con la regla @keyframes que veremos más adelante*) que queremos asociar al elemento HTML donde indicamos la propiedad.

Número de repeticiones

La propiedad animation-iteration-count permite indicar el número de veces que se repite la animación, pudiendo establecer un número concreto de repeticiones o indicando infinite para que se repita continuamente.

Dirección de la animación

Podemos indicar el orden en el que se reproducirán los fotogramas.

Valor	Significado
normal	Los fotogramas se reproducen en orden: desde el primero hasta el último.
reverse	Los fotogramas se reproducen en orden inverso: desde el final hasta el primero.
alternate	En iteraciones par, se reproducen como normal. En impares, como reverse.
alternate-reverse	En iteraciones par, se reproducen como reverse. En impares, como normal.

Modo completado

Mediante la propiedad animation-fill-mode podemos indicar que debe hacer la animación cuando no se está reproduciendo:

- El valor none realiza el comportamiento indicado en el párrafo anterior.
- El valor backwards indica que la animación debe tener aplicados los estilos del fotograma inicial antes de empezar.
- El valor forwards indica que la animación debe tener aplicados los estilos del fotograma final al terminar.
- El valor both indica que debe aplicar los dos casos anteriores (*backwards y forwards*).

Estados de animación

Por último, la propiedad animation-play-state nos permite establecer la animación a estado de reproducción running o pausarla mediante el valor *paused*. Esto en CSS no da demasiadas posibilidades, pero puede ser muy útil combinado con algo de Javascript.

Atajos...

CSS ofrece la posibilidad de resumir todas estas propiedades en una sola, para hacer nuestras hojas de estilos más compactas. El orden recomendado para los valores de la propiedad de atajo sería el siguiente:

```
div {
 /* animation: <name> <duration> <timing-function> <delay>
          <iteration-count> <direction> <fill-mode> <play-state> */

 animation: change-color 5s linear 0.5s 4 normal forwards running;
}
```

Regla @keyframe

Una animación esta formada por varios fotogramas, una secuencia de imágenes (*30-60 fotogramas por segundo, por ejemplo*) que mostradas una detrás de otra generan el efecto de movimiento que conocemos de una animación. En CSS, los fotogramas se crean a partir de propiedades CSS, y no hace falta definir tantos fotogramas. Sólo crearemos fotogramas clave y el resto de fotogramas los generará el navegador.

Para definir esos fotogramas clave, utilizaremos la regla @keyframes, la cuál es muy sencilla de utilizar.

Selectores from to

Las partes principales por las que debemos comenzar son:

Parte	Descripción
@keyframes	Regla para darle un nombre y definir los fotogramas clave de una animación.
from	Fotograma clave inicial con los estilos CSS a aplicar. Equivalente a 0%.
to	Fotograma clave final con los estilos CSS a aplicar. Equivalente a 100%.
porcentaje	Porcentaje específico de la animación con los estilos CSS a aplicar. Permite decimales.

```
@keyframes change-color {
    from { background: red; } /* Primer fotograma */
    to { background: green; } /* Segundo y último fotograma */
}
```

Selectores porcentuales

Los selectores from y to son muy similares a colocar 0% y 100%, así que los modificaremos y de esta forma podremos ir añadiendo nuevos fotogramas intermedios. Vamos a añadir un fotograma intermedio.

Esto, en código, se vería así:

```
@keyframes change-color {
 0% {
  background: red;       /* Primer fotograma */
 }
 50% {
  background: green;     /* Segundo fotograma */
 }
 100% {
  background: red;      /* Último fotograma */
 }
}
```

Funciones de tiempo

Tanto cuando hablamos de transiciones como cuando hablamos de animaciones tenemos dos propiedades que se aplican exactamente igual y se encargan de definir el ritmo o transcurso de la animación o transición en cuestión.

Propiedades	Significado
transition-timing-function	Ritmo de la transición
animation-timing-function	Ritmo de la animación

Enlaces de interés

Los siguientes enlaces que se facilitan son de sitios webs de confianza en los que podremos encontrar recursos e información más que interesante que nos ayudarán tanto a buscar información y ampliar lo aprendido en el curso, como a poder confirmar que estamos realizando bien el trabajo y podremos compartir los proyectos que vayamos haciendo.

URL's de referencia / Documentación

Acerca de transformaciones y animaciones CSS: https://lenguajecss.com/css/animaciones/transiciones/

Herramientas curiosas

Herramienta para ver y generar transformaciones CSS: https://css-transform.moro.es/
Herramienta para ver y generar transformaciones CSS:
https://www.toptal.com/developers/css3maker/css3-transform
Codigos de spinners realizados solo con HTML y CSS: https://cssloaders.github.io/
Para crear / generar el CSS de un degradado: https://cssgradient.io/