UT4: Usabilidad.
Transiciones y animaciones en CSS

DIW

Transiciones

Las transiciones en CSS nos permiten aplicar un cambio de estilo gradual a los elementos del documento HTML.

Ofrecen la ventaja de poder especificar el tiempo para que se produzca la transformación entre estilos, de esta forma podríamos utilizarlas para dar un efecto de animación.

Al utilizar una pseudoclase como *:hover*, el cambio de estilos ocurría de golpe, pasando de un estado inicial a otro final. Utilizando transiciones tenemos a nuestra disposición una gran flexibilidad que nos permitirá dotar de atractivos efectos de transición que harán que nuestros diseños sean más elegantes y agradables.

Transiciones

La especificación de las transiciones CSS todavía se encuentra en fase experimental.

Prefijo	Navegador
-moz-	Firefox
-webkit-	Safari y Chrome
-o-	Opera
-khtml-	Konqueror
-ms-	Internet Explorer y Microsoft Edge

Comprobar la compatibilidad.

Transiciones

Todos los parámetros para aplicar las transiciones se pueden establecer en una sola línea y también mediante propiedades por separado.

Veamos cómo se implementa en una sola línea.

```
transition:[propiedad a modificar] [Duración] [Tipo de animación] [Retardo];
```

Transiciones

Propiedad a modificar

Algunas de las propiedades que podemos modificar utilizando transiciones son las que se muestran en la siguiente lista:

- background-color
- border
- border-radius
- color
- top
- bottom
- left
- right
- box-shadow
- width
- height
- line-height
- margin
- opacity
- word-spacing
- letter-spacing
- fill
- padding
- stroke
- text-shadow
- vertical-align
- visibility
- z-index

Transiciones

Duración

Se debe especificar el número de segundos que va a durar la transición. Por ejemplo: 2s (2 segundos).

Retardo

Tiempo (en segundos) que el navegador esperará antes de poner en marcha la transición. Se especifica el número de segundos a esperar seguido de la «s» (ejemplo: 1s)

Transiciones

Tipo de animación

Esta propiedad es opcional y sirve para indicar la velocidad, el ritmo de la animación. Algunas de las posibilidades son las siguientes:

- **linear**: la velocidad de la animación es uniforme en todo el recorrido.
- **ease:** la velocidad se acelera al inicio, luego se retarda un poco y se acelera al final de nuevo.
- **ease-in**: la animación empieza lentamente y va aumentando progresivamente.
- **ease-out**: la animación empieza muy rápida y va descendiendo progresivamente.
- **ease-in-out**: la animación empieza y acaba lentamente, y es en la parte central del recorrido donde la velocidad es más rápida.

Transiciones

➨ Tipo de animación

Valor	Inicio	Transcurso	Final	Equivalente en cubic-beizer
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

Cubic-bezier

Transiciones

Se pueden controlar los componentes individuales de la transición usando las siguientes subpropiedades:

- **transition-property**: especifica el nombre o nombres de las propiedades CSS a las que deberían aplicarse las transiciones.
- **transition-duration:** especifica la duración en la que sucederán las transiciones.
- **transition-timing-function**: especifica la curva cúbica bézier que se usa para definir cómo se computan los valores intermedios para las propiedades.
- **transition-delay**: define el tiempo de espera entre el momento en que se cambia una propiedad y el inicio de la transición.

Transiciones

```css
div {
  transition-property: opacity, left, top, height;
  transition-duration: 3s, 5s;
}
```

```css
div {
  transition-property: opacity, left, top, height;
  transition-duration: 3s, 5s, 3s, 5s;
}
```

```css
div {
  transition-property: opacity, left;
  transition-duration: 3s, 5s, 2s, 1s;
}
```

```css
div {
  transition-property: opacity, left;
  transition-duration: 3s, 5s;
}
```

Transformaciones

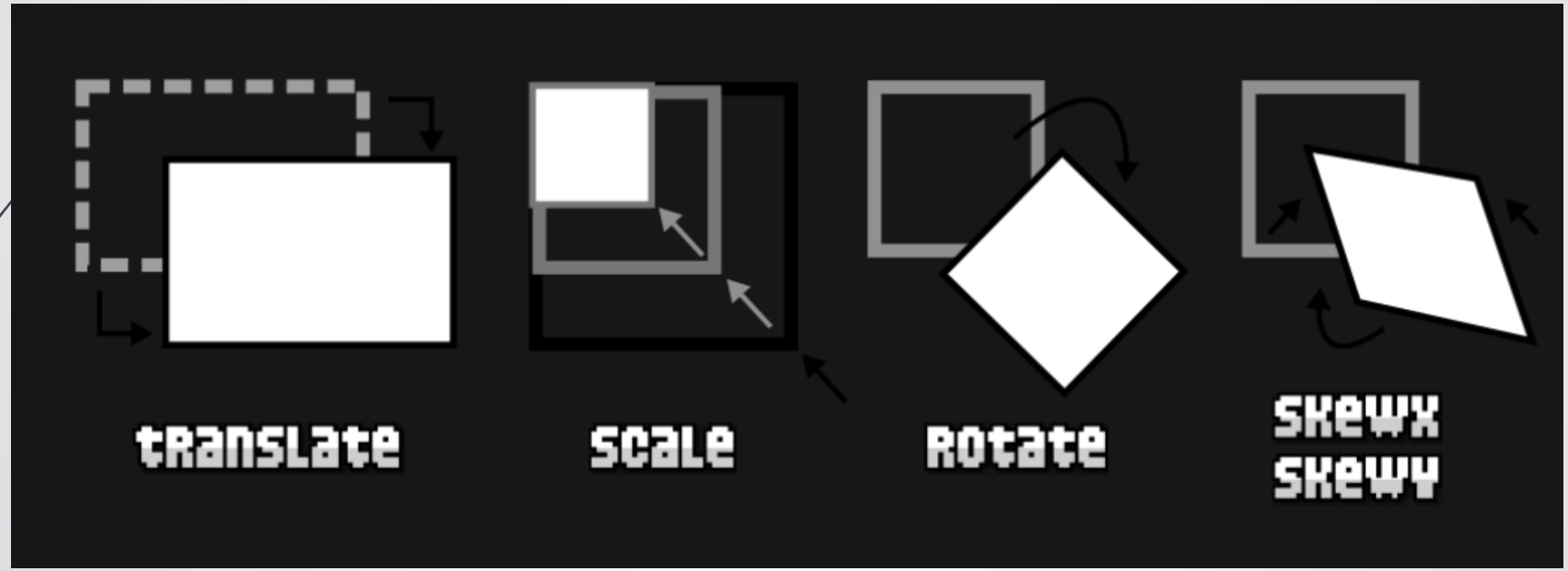
Las transformaciones son una de las características de CSS más interesantes y potentes que se introducen en el lenguaje para convertir las hojas de estilo en un sistema capaz de realizar efectos visuales 2D y 3D.

Con ellas podemos hacer cosas como mover elementos, rotarlos, aumentarlos o disminuirlos y otras transformaciones relacionadas o combinadas.

Estas transformaciones se pueden efectuar en CSS mediante la propiedad **transform** que permite recibir una función de transformación determinada, la cuál será aplicada en el elemento HTML en cuestión seleccionado mediante CSS.

Transformaciones

Podemos realizar las siguientes transformaciones:



Transformaciones

Funciones	Significado
`translateX(x)`	Traslada el elemento una distancia de SIZE x horizontalmente.
`translateY(y)`	Traslada el elemento una distancia de SIZE y verticalmente.
`translate(x, y)`	Propiedad de **atajo** de las dos anteriores.
`translate(x)`	Equivalente a `translate(x, 0)`
`translateZ(z)`	Ver en Transformaciones 3D
`translate3d(x, y, z)`	Ver en Transformaciones 3D

Transformaciones

Funciones	Significado
`rotateX(x)`	Establece una rotación 2D en ANGLE x sólo para el eje horizontal X.
`rotateY(y)`	Establece una rotación 2D en ANGLE y sólo para el eje vertical Y.
`rotateZ(z)`	Establece una rotación 2D en ANGLE z sobre si mismo.
`rotate(z)`	Alias a la anterior.
`rotate3d(x, y, z, a)`	Ver en Transformaciones 3D

Transformaciones

Funciones	Significado
`scaleX(fx)`	Reescala el elemento un número de NUMBER `fx` veces (sólo en horizontal).
`scaleY(fy)`	Reescala el elemento un número de NUMBER `fy` veces (sólo en vertical).
`scale(fx, fy)`	Propiedad de **atajo** de las dos anteriores (escalado simétrico).
`scale(fx)`	Equivalente al anterior: `scale(fx, fx)`.
`scaleZ(fz)`	Ver en Transformaciones 3D
`scale3D(fx, fy, fz)`	Ver en Transformaciones 3D

Transformaciones

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

```
.element {
  width: 50px;
  height: 50px;
  background: grey;
  transform: translate(150px, 100px) rotate(25deg);
}
```

Animaciones

Las animaciones CSS amplían el concepto de transiciones, convirtiéndolo en algo mucho más flexible y potente, en el que no es necesario que el usuario interactúe de alguna forma (como pasa en las transiciones).

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.

Animaciones

Para crear animaciones CSS es necesario realizar 2 pasos:

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

Animaciones

- Propiedades relacionadas con las animaciones

Propiedades	Descripción	Valor
animation-name	Nombre de la animación a aplicar.	**none** | nombre
animation-duration	Duración de la animación.	**0** | TIME
animation-timing-function	Ritmo de la animación.	Ver funciones de tiempo
animation-delay	Retardo en iniciar la animación.	**0** | TIME
animation-iteration-count	Número de veces que se repetirá.	**1** | infinite | NUMBER
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
animation-composition	Como se «mezclan» varias animaciones.	**replace** | add | accumulate
animation-timeline	Define una línea de tiempo animada.	Ver animaciones de scroll

Animaciones

Definiendo nuestra primera animación

- **animation-name**: es la propiedad principal de las animaciones. 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.
- **animation-duration**: igual que transition-duration.
- **animation-delay**: igual que transition-delay.
- **animation-timing-function**: establece el ritmo de la animación de forma idéntica a como lo hacía transition-timing-function.
- **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.

Animaciones

Definiendo nuestra primera animación

- **animation-direction**: indica el orden en el que se reproducirán los fotogramas, pudiendo escoger un valor entre los siguientes:
  - normal: se reproducen en orden.
  - reverse: se reproducen desde el final hasta el inicio, en orden inverso.
  - alternate: las iteraciones par se reproducen normal, y las impares como reverse.
  - alternate-reverse: las iteraciones par se reproducen reverse, las impares como normal.
- **animation-fill-mode**: podemos indicar que debe hacer la animación cuando no se está reproduciendo: (none por defecto)
  - backwards: la animación tiene los estilos del fotograma inicial antes de empezar.
  - forwards: en este caso tiene aplicados los estilos del fotograma del final al terminar.
  - both: aplica los dos casos anteriores.

Animaciones

Definiendo nuestra primera animación

- **animation-play-state**: nos permite establecer la animación a estado de reproducción *running* o pausarla mediante el valor *paused*.

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:

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

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

Animaciones

➡ Definiendo nuestra primera animación

Es posible encadenar animaciones utilizando animaciones múltiples combinadas con la propiedad ***animation-delay***

```
.animated {
  animation:
    move-right 5s linear 0s,    /* Comienza a los 0s (no hay anterior) */
    look-up 2.5s linear 5s,     /* Comienza a los 5s (5 de la anterior) */
    move-left 5s linear 7.5s,   /* Comienza a los 7.5s (5 + 2.5 de la anterior) */
    dissapear 2s linear 9.5s;   /* Comienza a los 9.5s (5 + 2.5 + 2 de la anterior) */
}
```

Animaciones

Definiendo nuestra primera animación

- **animation-composition**: podemos indicar al navegador como se van a aplicar por parte del navegador múltiples animaciones. Los valores disponibles a utilizar son los siguientes:
  - replace: las animaciones se sobreescriben. Valor por defecto.
  - add: las animaciones se añaden.
  - accumulate: las animaciones se acumulan.
  - alternate-reverse: las iteraciones par se reproducen reverse, las impares como normal.

Ejemplo: partiendo de un cuadrado de 200px por 200px, color azul, aplica una animación llamada move que translade 1000 px el cuadrado horizontalmente hacia la derecha durante 6 segundos, luego que se aplique otra animación, llamada giro, que rote 180º durante otros 6 segundos y reduzca su tamaño a la mitad. A partir de ahí deben repetirse simultáneamente. Cada animación también debe cambiar el color del elemento.

Animaciones

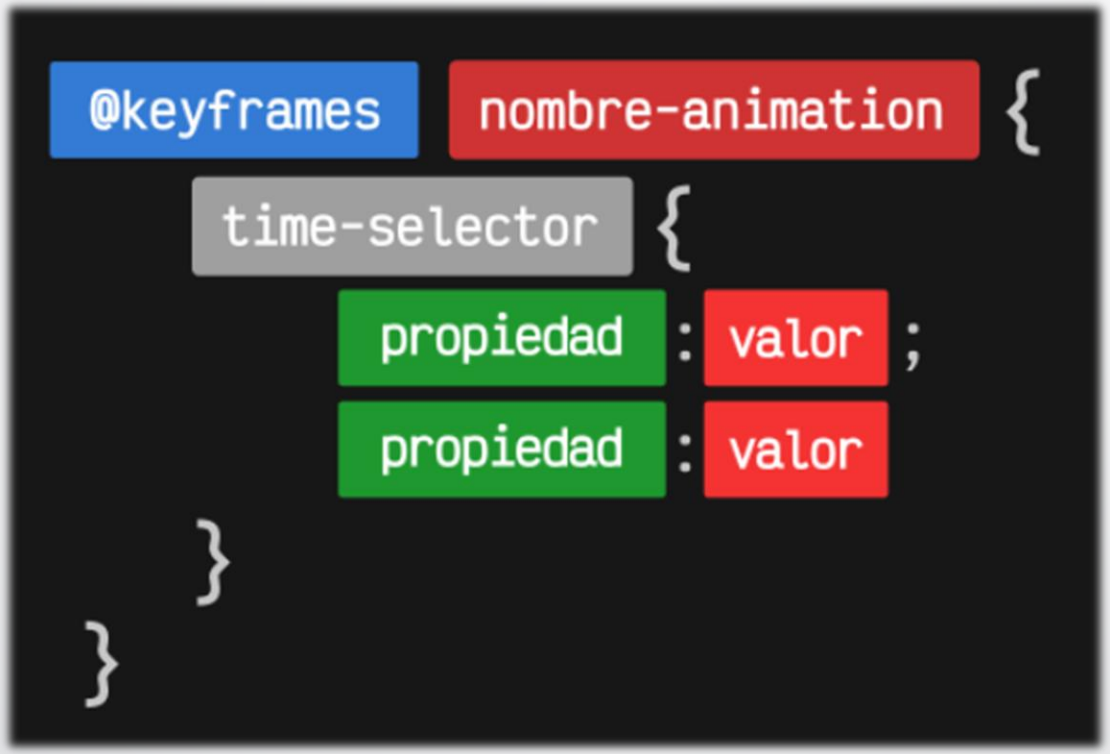
Una vez configurada la propiedad animation, nos falta la parte donde tenemos que definir los fotogramas de la animación.

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.

Animaciones

La regla @Keyframes

Para definir esos fotogramas clave, utilizaremos la regla @keyframes, la cuál es muy sencilla de utilizar. Se basa en el siguiente esquema:



Animaciones

La regla @Keyframes

Cada uno de los **time-selector** define un momento clave de cada uno de los fotogramas clave de nuestra animación. Veremos que pueden definirse muchos en una misma animación

- **Selectores from y to:** usando estos selectores definimos
  - from: fotograma clave inicial con los estilos CSS a aplicar
  - to: fotograma clave final con los estilos CSS a aplicar
- **Selectores porcentuales:** definiendo porcentajes podremos ir añadiendo nuevos fotogramas intermedios. Colocando 0% y 100% simularíamos un comportamiento similar a usar los selectores from y to.

Animaciones

La regla @Keyframes

Podemos combinarlos en una misma animación

```
@keyframes animLogo {
    from {
        transform: rotateX(0deg);
    }

    50% {
        transform: rotateX(180deg);
    }

    to {
        transform: rotateX(360deg);
    }
}
```

Animaciones

- La regla @Keyframes
  - **Animar personajes**: en este enlace podemos ver paso a paso como realizar el proceso.

    Las funciones de salto permiten establecer una transición o animación en un número concreto de pasos muy específico, que se pasa por parámetro n. Monkey Island.

  - **View Transition**: para implementar transiciones entre páginas.
  - **Timeline**: líneas de tiempo que permiten realizar animaciones de un elemento contenedor, controlando su progreso mediante el scroll, o sólo cuando el elemento está a la vista en una determinada región visible.