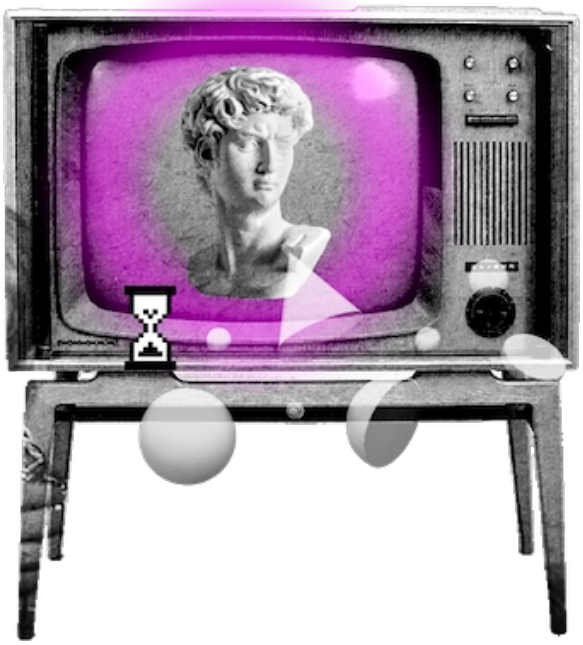
NUCBA

DISEÑO WEB

ANIMACIONES Y MICROINTERACCIONES



APUNTES

CSS VI - Animaciones y transiciones

Concepto de Ejes

El concepto de **ejes** es muy importante para comprender **cómo los elementos de una página web pueden moverse y cambiar de forma dinámica mediante código CSS**. Los ejes **son los puntos de referencia que se utilizan para determinar la dirección y la velocidad de los cambios que se aplican a un elemento**.

Hay dos tipos principales de ejes que se utilizan en CSS: **el eje X y el eje Y**. El eje X representa la **dirección horizontal**, mientras que el eje Y representa la **dirección vertical**. Juntos, estos dos ejes **forman un plano cartesiano** que se utiliza para representar la posición de un elemento en una página web.

Cuando se aplican animaciones y transiciones a un elemento utilizando CSS, se pueden definir los ejes de movimiento para controlar la dirección y la velocidad del cambio. Por ejemplo, se puede utilizar el eje X para **mover un elemento de izquierda a derecha**, o el eje Y **para moverlo de arriba a abajo**.

Los ejes también pueden ser utilizados para **controlar la rotación y la escala de un elemento**. En estos casos, **se utilizan los ejes Z y XYZ** para **representar la dirección en la que el elemento está siendo transformado**.

Transformaciones

Las transformaciones en CSS son una herramienta útil que **nos permite cambiar la apariencia y posición de elementos HTML sin modificar su estructura**. Con las transformaciones, podemos girar, mover, escalar y deformar elementos de manera dinámica y fluida.

Existen diferentes tipos de transformaciones que se pueden aplicar a los elementos HTML, veamos las más importantes.

Translate

Esta transformación nos permite mover un elemento en diferentes direcciones (izquierda, derecha, arriba o abajo) a través del eje X e Y. La sintaxis para aplicar esta transformación es:

```
transform: translate(x, y);
```

Donde **x** e **y** son valores numéricos que indican cuánto deberá moverse un elemento en cada uno de esos ejes.

Scale

La transformación Scale nos **permite aumentar o disminuir el tamaño de un elemento HTML**. Podemos escalar en ambas direcciones (X e Y) o sólo en una de ellas. La sintaxis para aplicar esta transformación es:

```
transform: scale(x, y);
```

Donde **x** e **y** son valores numéricos que indican cuánto deberá crecer un elemento en cada uno de esos ejes.

Skew

La transformación Skew nos **permite inclinar o deformar un elemento HTML**. La sintaxis para aplicar esta transformación es:

```
transform: skew(x, y);
```

Donde **x** e **y** son valores numéricos que indican cuánto se deformará un elemento en cada uno de esos ejes.

Rotate

Esta transformación nos permite **girar un elemento HTML en un ángulo determinado**. La sintaxis para aplicar esta transformación es:

```
transform: rotate(angulo);
```

Donde el ángulo es un valor en grados (**deg**) que indica la rotación.

Transiciones en CSS

Las **transiciones** son **una técnica de animación en CSS que permite cambiar gradualmente el valor de una propiedad de un estado inicial a un estado final**. Esto significa que podemos crear efectos de animación en los elementos HTML sin necesidad de utilizar JavaScript ni plugins adicionales.

Las mismas se manejan mediante el uso de varias propiedades que vamos a ver a continuación.

Transition-property

La propiedad **transition-property** en CSS se **utiliza para especificar los nombres de las propiedades que deseas animar durante una transición**. Es decir, indica qué propiedad o propiedades queremos que cambien gradualmente a lo largo de la duración de la transición.

Por ejemplo, si queremos que el color de fondo y el color del texto cambien de forma suave al pasar el ratón por encima de un enlace, podemos usar la propiedad transition-property para especificar que queremos animar las propiedades **background-color y color**.

La sintaxis de la propiedad transition-property es bastante sencilla. Solo necesitamos escribir el nombre o los nombres de las propiedades CSS que queremos animar, separadas por comas. Por ejemplo:

```css
a:hover {
  transition-property: background-color, color;
}
```

En caso de que quisieramos aplicar la transición a todas las propiedades que cambiarán, podemos utilizar la palabra clave **“all”**.

Transition-duration

La propiedad **transition-duration** es utilizada en CSS para **establecer la duración de una transición que se aplica a un elemento**. Esta propiedad define el tiempo que tardará el cambio de un estado al otro, lo que permite una animación suave y gradual.

La sintaxis básica de la propiedad transition-duration es la siguiente:

```
transition-duration: valor;
```

Donde "valor" puede ser cualquier valor numérico seguido de una unidad de tiempo, como segundos (s) o milisegundos (ms). Por ejemplo, para establecer una duración de transición de 2 segundos, se puede escribir:

```
transition-duration: 2s;
```

Transition-timing-function

La propiedad **transition-timing-function** en CSS **define cómo se calculan los valores intermedios durante una transición, es decir, cómo se va a realizar la transición entre un estado inicial y un estado final**. Esta propiedad acepta diferentes valores para definir la forma en que se realizará la transición, algunos de ellos son:

ease: Es el valor por defecto y hace que la transición se realice de forma suave, con una aceleración y desaceleración gradual.
linear: Hace que la transición sea lineal, es decir, que la velocidad de la transición sea constante a lo largo de todo el proceso.
ease-in: Hace que la transición empiece de forma suave y vaya acelerando a medida que se va desarrollando.
ease-out: Hace que la transición empiece de forma rápida y vaya disminuyendo la velocidad a medida que se va acercando al final.
ease-in-out: Hace que la transición empiece de forma suave, acelere en la mitad del proceso y luego desacelere al final.

Además, se pueden usar funciones matemáticas como **cubic-bezier()** para crear una transición personalizada, definiendo los puntos de control de la curva de aceleración.

```
transition-timing-function: ease-in-out;
```

Transition-delay

La propiedad **transition-delay** permite **especificar el tiempo que debe esperar una transición antes de que comience**. Es decir, puedes establecer un retraso para que una transición comience después de un determinado período de tiempo.

La sintaxis para **transition-delay** es la siguiente:

```
transition-delay: <tiempo>;
```

Donde **<tiempo>** es un valor en **segundos(s)** o **milisegundos(ms)**.

Transition

La propiedad **transition** en CSS es **una forma abreviada (shorthand) de definir las cuatro propiedades relacionadas con las transiciones**: **transition-property, transition-duration, transition-timing-function y transition-delay**. Esta propiedad abreviada permite definir todas las propiedades de transición en una sola declaración.

La sintaxis de la propiedad transition es la siguiente:

```
transition: [property] [duration] [timing-function] [delay];
```

Cada valor dentro de los corchetes corresponde a una de las cuatro propiedades de transición.

Animaciones en CSS

Las **animaciones** en CSS **permiten crear efectos visuales en una página web al cambiar gradualmente los valores de las propiedades CSS de un elemento a lo largo del tiempo**. Se pueden animar una o varias propiedades a la vez, y la duración y el tipo de animación se pueden especificar con CSS.

Keyframes

Las animaciones se definen utilizando la regla **@keyframes seguida del nombre de la animación**. Luego, entre llaves, se especifican los valores que tendrán las propiedades que se quieren animar en los distintos puntos de la animación.

```
@keyframes nombreAnimacion  {
    /*Animación*/
}
```

Veamos un ejemplo en el cual definimos una animación en la cual el color de la letra del elemento al que se le aplicará la animación será distinto en 3 momentos distintos:

```
@keyframes nombreAnimacion {
  0% {
    color:white;
  }
  50% {
    color:red;
  }
  100% {
    color:blue
  }
}
```

Propiedades de las animaciones

Ahora que ya vimos cómo crearlas, debemos colocarlas en un elemento de nuestro HTML. Para esto, existen varias propiedades que se pueden aplicar. A continuación vamos a ver una por una.

Animation-name

La propiedad **animation-name** nos permite **definir el nombre de la propiedad que queremos aplicar** a un elemento. Es la única propiedad obligatoria a la hora de colocar una animación a un elemento.

```
animation-name: nombreAnimacion;
```

Animation-duration

La propiedad **animation-duration** nos permite **definir la duración de la animación**, especificada en segundos(s) o milisegundos(ms).

```
animation-duration: tiempo; /* Ej: 2s ,9ms, etc.
```

Animation-timing-function

La propiedad **animation-timing-function** nos permite **definir la curva de velocidad de la animación**. Puede ser **ease(predeterminada),linear,ease-in,ease-out** o **ease-in-out** (valores que ya vimos con la propiedad transition-timing-function)**.**

```
animation-timng-function: curvaDeVelocidad;
```

Animation-delay

La propiedad **animation-delay** nos permite **especificar el tiempo que debe esperar una animación antes de que comience**, especificada en segundos(s) o milisegundos(ms).

```
animation-delay: tiempo;
```

Animation-iteration-count

La propiedad **animation-iteration-count** permite **indicar el número de veces que se debe repetir una animación**. Puede ser un número determinado de veces, **infinite**, para repetir la animación infinitamente, o **alternate** , para repetir la animación de ida y vuelta.

```
animation-iteration-count: nroVeces;
```

Animation-direction

La propiedad **animation-direction determina la dirección en la que debe reproducirse la animación.** Su valor puede ser **normal(predeterminado)**, **reverse(del final al principio), alternate (de principio a fin y viceversa) o alternate-reverse (de fin a principio y viceversa).**

```
animation-direction: direccion;
```

Animation-fill-mode

La propiedad **animation-fill-mode** permite **especificar cómo deben ser los valores de las propiedades antes y después de la animación.** Estos valores pueden ser:

none: el elemento mantiene su estilo original antes y después de la animación.
forwards: el elemento mantiene el último estilo de la animación después de que se complete.
backwards: el elemento toma el primer estilo de la animación antes de que comience.
both: el elemento aplica tanto forwards como backwards.

```
animation-fill-mode: estilo;
```

Animation-play-state

La propiedad **animation-play-state** permite **pausar y reanudar una animación en CSS sin tener que detenerla y volver a iniciarla**. Sus valores pueden ser **running** o **paused.**

```
animation-play-state: state;
```

Esta propiedad normalmente se usa de manera dinámica, pudiendo alterar su valor para parar y reanudar una animación desde Javascript.

Animation

La propiedad **animation** es un shorthand que nos permite definir varias propiedades de una animación en una misma línea de código.
Es importante recordar que **la única propiedad que es obligatoria es la propiedad name.** El resto, pueden colocarse o no, en el siguiente orden:

```
animation: name duration timing-function delay iteration-count
direction fill-mode play-state;
```

Nucba tip: *Sigan investigando sobre los temas aquí expuestos. Practiquen, apoyense en la documentación oficial y sobre todo mantengan la constancia y la curiosidad a medida que vayan aprendiendo cosas nuevas.*

#HappyCoding 🚀