Diplomatura

FullStack Developer

CSS
Posición relativa y absoluta, pseudoclases, pseudoselementos y transiciones

Posición relativa y absoluta

En CSS, la propiedad position es fundamental para controlar el flujo y la disposición de los elementos en una página web. Hay varias opciones disponibles que afectan cómo un elemento se coloca en relación con otros y su contenedor. Aquí te explico los diferentes valores de la propiedad position:

1. static (por defecto)
•Es el valor por defecto para todos los elementos. Los elementos con position: static están en el flujo natural del documento, sin cambiar su posición habitual en la página.
•No responde a las propiedades top, right, bottom o left.

2. relative
•El elemento se posiciona en relación con su ubicación original en el flujo del documento.
•Las propiedades top, right, bottom, y left desplazan el elemento desde su posición original, pero sigue ocupando su espacio original en el flujo del documento.

```css
.elemento {
    position: relative;
    top: 10px;
    left: 20px;
}
```

En este caso, el elemento se moverá 10 píxeles hacia abajo y 20 píxeles hacia la derecha desde su posición original.

3. absolute

•El elemento se posiciona en relación con el primer contenedor padre que tenga una posición distinta a static (es decir, un contenedor con relative, absolute o fixed). Si no hay un contenedor padre posicionado, el elemento se posiciona respecto al <html>.
•El elemento se saca del flujo del documento, lo que significa que no ocupará espacio, y los elementos cercanos se comportarán como si no existiera.

```css
.elemento {
    position: absolute;
    top: 50px;
    left: 30px;
}
```

4. fixed

•Similar a absolute, pero el elemento se posiciona en relación con la ventana del navegador en lugar del contenedor padre.

•Se mantiene fijo en su lugar incluso cuando se hace scroll en la página.

```
.elemento {
    position: fixed;
    bottom: 10px;
    right: 10px;
}
```

Esto coloca al elemento en la esquina inferior derecha de la pantalla y permanece allí aunque se despliegue la página.

5. sticky

•Es una combinación de relative y fixed. El elemento se comporta como relative hasta que se alcanza un punto específico en el scroll, momento en el cual se comporta como fixed.

```
.elemento {
    position: sticky;
    top: 0;
}
```

Esto hace que el elemento quede "pegajoso" y se quede en la parte superior de la pantalla cuando se hace scroll hasta cierto punto.

Pseudoclases

Las pseudoclases se utilizan para aplicar estilos a elementos HTML en función de su estado o posición en relación a otros elementos. Se definen añadiendo dos puntos : seguidos del nombre de la pseudoclase después del selector al que se desea aplicar. Permiten dirigir los estilos CSS a elementos específicos que cumplen ciertas condiciones, sin necesidad de modificar la estructura HTML. La sintaxis general es: un selector, seguido de dos puntos y el nombre de la pseudoclase, y entre llaves {} las declaraciones de estilo correspondientes.

```
selector:pseudoclase {
    propiedad: valor;
    propiedad: valor;
    ...
}
```

A diferencia de las clases regulares, las pseudoclases no se definen en el HTML, sino que se usan directamente en las reglas CSS. Estas pseudoclases se indican con dos puntos : seguidos del nombre de la pseudoclase después del selector.

- **:link**: Aplica estilos a los enlaces que aún no han sido visitados.
- **:visited**: Aplica estilos a los enlaces que ya han sido visitados.
- **:hover**: Aplica estilos cuando el cursor del mouse se posiciona sobre un elemento.
- **:active**: Aplica estilos cuando un elemento está siendo activado (por ejemplo, al hacer clic en un botón).
- **:focus**: Aplica estilos cuando un elemento tiene el foco (por ejemplo, cuando se hace clic en un campo de entrada de texto).
- **:first-child**: Aplica estilos al primer hijo de un elemento padre.
- **:last-child**: Aplica estilos al último hijo de un elemento padre.
- **:nth-child(n)**: Aplica estilos a hijos específicos de un elemento padre, donde n es un número, una palabra clave o una fórmula.
- **:nth-last-child(n)**: Similar a :nth-child(n), pero cuenta desde el último hijo hacia arriba.

- **:first-of-type**: Aplica estilos al primer elemento de su tipo entre los hijos de su elemento padre.
- **:last-of-type**: Aplica estilos al último elemento de su tipo entre los hijos de su elemento padre.
- **:nth-of-type(n)**: Aplica estilos a elementos específicos de su tipo entre los hijos de su elemento padre.
- **:nth-last-of-type(n)**: Similar a :nth-of-type(n), pero cuenta desde el último elemento de su tipo hacia arriba.
- **:only-child**: Aplica estilos a un elemento que es el único hijo de su elemento padre.
- **:only-of-type**: Aplica estilos a un elemento que es el único de su tipo entre los hijos de su elemento padre.
- **:empty**: Aplica estilos a elementos que no tienen hijos (incluyendo texto y espacios en blanco).

- **:checked**: Aplica estilos a elementos de entrada (como casillas de verificación o botones de radio) cuando están seleccionados.
- **:disabled**: Aplica estilos a elementos de formulario deshabilitados.
- **:enabled**: Aplica estilos a elementos de formulario habilitados.
- **:target**: Aplica estilos a un elemento que es el objetivo de un enlace interno (con un fragmento de URL).
- **:root**: Aplica estilos al elemento raíz del documento, que suele ser el elemento <html>. Es útil para definir variables CSS.
- **:valid**: Aplica estilos a los elementos de formulario que contienen datos válidos.
- **:invalid**: Aplica estilos a los elementos de formulario que contienen datos inválidos.
-
- **:required**: Aplica estilos a los campos de formulario que son obligatorios.
- **:in-range**: Aplica estilos a los elementos <input> cuyo valor se encuentra dentro de un rango válido.

```html
  <h1>Bienvenido</h1>
  <p>Aquí hay algunos <a href="#">enlaces</a>.</p>
  <ul>
    <li>Primer elemento</li>
    <li>Segundo elemento</li>
    <li>Tercer elemento</li>
  </ul>
  <input type="text" placeholder="Ingrese su nombre">
  <input type="checkbox" id="opcion"><label
for="opcion">Opción seleccionada</label>
  <table>
    <tr>
      <th>Nombre</th>
      <th>Edad</th>
    </tr>
    <tr>
      <td>Juan</td>
      <td>25</td>
    </tr>
    <tr>
      <td>María</td>
      <td>30</td>
    </tr>
    <tr>
      <td>Pedro</td>
      <td>40</td>
    </tr>
  </table>
```

```css
/* 1. :hover */
a:hover {
    color: red;
}

/* 2. :first-child */
ul>li:first-child {
    font-weight: bold;
}

/* 3. :focus */
input:focus {
    background-color: #f5f5f5;
}

/* 4. :checked */
input[type="checkbox"]:checked+label {
    color: green;
}

/* 5. :nth-child(n) */
tr:nth-child(even) {
    background-color: #f2f2f2;
}
```

Pseudoelementos

Los pseudoelementos en CSS son una forma de seleccionar y aplicar estilos a partes específicas de un elemento HTML. A diferencia de las pseudoclases, que seleccionan elementos basados en un estado o condición, los pseudoelementos seleccionan una parte concreta de un elemento. Los pseudoelementos se denotan utilizando dos dos puntos seguidos del nombre del pseudoelemento. Por ejemplo: ::before o ::after.

```
a::before {
    content: "🔗 ";
}

a::after {
    content: " ⬅️";
}
```

En este caso, se añadirá un ícono de enlace antes de cada enlace <a> y una flecha hacia la izquierda después de cada enlace. Los pseudoelementos son muy útiles para añadir contenido decorativo o funcional a los elementos HTML sin necesidad de modificar el HTML directamente. También permiten aplicar estilos específicos a partes concretas de un elemento, como la primera línea o letra de un texto.

- Algunos de los pseudoelementos más comunes son:
- **::before**: Crea un pseudo-elemento que es el primer hijo del elemento seleccionado. Comúnmente se usa para añadir contenido antes del contenido real del elemento.
- **::after**: Crea un pseudo-elemento que es el último hijo del elemento seleccionado. Comúnmente se usa para añadir contenido después del contenido real del elemento.
- **::first-line**: Aplica estilos a la primera línea de texto de un elemento.
- **::first-letter**: Aplica estilos a la primera letra de un elemento.
- **::selection**: Aplica estilos al texto seleccionado por el usuario.
- **::placeholder**: Aplica estilos al texto de marcador de posición (placeholder) en un elemento de entrada de formulario.

```html
<article>
    <p>
        Lorem ipsum dolor sit amet.
    </p>
</article>
```

```css
article>p::selection{
    background-color: red;
    color: rgb(255, 255, 255);
}
```

Este sería el resultado de aplicar este pseudoelemento ::selection

Animaciones

Las animaciones en CSS permiten animar la transición entre diferentes estados de un elemento HTML. Esto significa que se pueden cambiar propiedades como el color, el tamaño, la posición, la opacidad, etc., de una manera suave y animada en lugar de un cambio abrupto.Las animaciones se definen mediante las reglas @keyframes y la propiedad animation.

```
@keyframes mover {
    0% {
        transform: translateX(0);
    }

    50% {
        transform: translateX(100px);
    }

    100% {
        transform: translateX(0);
    }
}
```

```
div {
     width: 100px;
     height: 100px;
     background-color: red;
     animation-name: mover;
     animation-duration: 2s;
     animation-iteration-count: infinite;
 }
```

Aquí, primero definimos una secuencia de animación llamada mover con la regla @keyframes. Dentro de esta regla, definimos los estilos que se aplicarán en diferentes puntos clave (keyframes) de la animación, expresados en porcentajes.

En este ejemplo, la animación comienza con la caja en su posición original (translateX(0)), luego se mueve 100 píxeles a la derecha (translateX(100px)) en el 50% de la animación, y finalmente vuelve a su posición original (translateX(0)) al final de la animación.

Luego, aplicamos esta animación a un elemento div utilizando las propiedades animation-name, animation-duration y animation-iteration-count. La duración se establece en 2 segundos, y se repite infinitamente (infinite).

Otras propiedades importantes de animación son:

- animation-delay: Retrasa el inicio de la animación.animation-timing-function: Controla la curva de velocidad de la animación (lineal, aceleración, desaceleración, etc.).

- animation-direction: Indica si la animación debe repetirse en reversa en cada iteración.

- animation-fill-mode: Define cómo se aplican los estilos antes y después de la animación.

Las animaciones en CSS son muy útiles para crear interacciones y efectos visuales atractivos en una página web, como carruseles, transiciones de elementos, animaciones de carga, etc. Sin embargo, para animaciones más complejas o con un alto rendimiento, es recomendable utilizar tecnologías como JavaScript.

Y las transformaciones en CSS permiten alterar el aspecto de un elemento sin afectar su flujo en el documento.

Veamos una lista de posibilidades que tenemos.

Transformaciones 2D

1.translate(x, y)

- Mueve un elemento desde su posición actual a una nueva posición definida por las coordenadas x (horizontal) e y (vertical).
- Ejemplo: transform: translate(50px, 100px); (mueve el elemento 50px a la derecha y 100px hacia abajo).

2.translateX(x)

- Mueve un elemento a lo largo del eje X.
- Ejemplo: transform: translateX(50px); (mueve el elemento 50px a la derecha).

3.translateY(y)

- Mueve un elemento a lo largo del eje Y.
- Ejemplo: transform: translateY(100px); (mueve el elemento 100px hacia abajo).

4.scale(x, y)

- Escala el tamaño de un elemento en los ejes X e Y. Los valores son factores de escala.
- Ejemplo: transform: scale(1.5, 2); (aumenta el tamaño del elemento 1.5 veces en el eje X y 2 veces en el eje Y).

5.scaleX(x)

- Escala el tamaño de un elemento solo en el eje X.
- Ejemplo: transform: scaleX(1.5); (aumenta el tamaño del elemento 1.5 veces en el eje X).

6.scaleY(y)

- Escala el tamaño de un elemento solo en el eje Y.
- Ejemplo: transform: scaleY(2); (aumenta el tamaño del elemento 2 veces en el eje Y).

7.rotate(angle)

- Rota un elemento en torno a su punto de origen. El ángulo se especifica en grados (deg), radianes (rad), giros (turn), o grados por ciento (grad).
- Ejemplo: transform: rotate(45deg); (rota el elemento 45 grados en sentido horario).

8.skew(x-angle, y-angle)

- Deforma un elemento en los ejes X e Y por los ángulos especificados.
- Ejemplo: transform: skew(30deg, 20deg); (deforma el elemento 30 grados en el eje X y 20 grados en el eje Y).

9.skewX(angle)

- Deforma un elemento solo en el eje X.
- Ejemplo: transform: skewX(30deg); (deforma el elemento 30 grados en el eje X).

10.skewY(angle)

- Deforma un elemento solo en el eje Y.
- Ejemplo: transform: skewY(20deg); (deforma el elemento 20 grados en el eje Y).

Transformaciones 3D

1.translateZ(z)

- Mueve un elemento a lo largo del eje Z (profundidad).
- Ejemplo: transform: translateZ(50px); (mueve el elemento 50px hacia fuera de la pantalla).

2.translate3d(x, y, z)

- Mueve un elemento en el espacio 3D utilizando los ejes X, Y y Z.
- Ejemplo: transform: translate3d(50px, 100px, 200px); (mueve el elemento 50px a la derecha, 100px hacia abajo y 200px hacia fuera de la pantalla).

3.scaleZ(z)

- Escala el tamaño de un elemento en el eje Z.
- Ejemplo: transform: scaleZ(1.5); (aumenta el tamaño del elemento 1.5 veces en el eje Z).

4.scale3d(x, y, z)

- Escala el tamaño de un elemento en los ejes X, Y y Z.
- Ejemplo: transform: scale3d(1.5, 2, 1.2); (aumenta el tamaño del elemento 1.5 veces en el eje X, 2 veces en el eje Y y 1.2 veces en el eje Z).

5.rotateX(angle)

- Rota un elemento alrededor del eje X.
- Ejemplo: transform: rotateX(45deg); (rota el elemento 45 grados alrededor del eje X).

6.rotateY(angle)

- Rota un elemento alrededor del eje Y.
- Ejemplo: transform: rotateY(45deg); (rota el elemento 45 grados alrededor del eje Y).

7.rotateZ(angle)

- Rota un elemento alrededor del eje Z (equivalente a rotate(angle) en 2D).
- Ejemplo: transform: rotateZ(45deg); (rota el elemento 45 grados alrededor del eje Z).

8.rotate3d(x, y, z, angle)

- Rota un elemento alrededor de un vector definido por los valores x, y y z y el ángulo especificado.
- Ejemplo: transform: rotate3d(1, 1, 0, 45deg); (rota el elemento 45 grados alrededor del vector (1,1,0)).

9.perspective(n)

- Establece la profundidad de perspectiva para un elemento. Este valor afecta cómo se renderizan las transformaciones 3D.
- Ejemplo: transform: perspective(500px); (aplica una perspectiva de 500px).

Animaciones con transiciones

Las transiciones CSS son una forma sencilla de animar cambios de estado en un elemento, como cuando se pasa el cursor sobre él o se hace clic. A diferencia de las animaciones con @keyframes, las transiciones no requieren definir múltiples fotogramas, solo el estado inicial y final.

Las transiciones se controlan mediante las siguientes propiedades:

1. **transition-property:** Especifica la(s) propiedad(es) CSS que se van a animar. Puedes indicar una sola propiedad o múltiples separadas por coma.

```
transition-property: background-color, transform;
```

2. **transition-duration:** Establece la duración de la transición, es decir, el tiempo que tomará la animación desde el estado inicial hasta el final.

```
transition-duration: 0.5s; /* 0.5 segundos */
```

3. **transition-timing-function:** Define la curva de aceleración de la transición, al igual que animation-timing-function en las animaciones por @keyframes.

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

4. **transition-delay:** Agrega un retraso antes de que comience la transición.

```
transition-delay: 0.2s; /* 0.2 segundos de retraso */
```

Estas cuatro propiedades también se pueden abreviar en una sola, siguiendo el orden:
transition: property duration timing-function delay;

```
transition: background-color 0.5s ease 0.2s, transform 1s linear;
```

Un uso común de las transiciones es animar cambios de estado al pasar el cursor sobre un elemento:

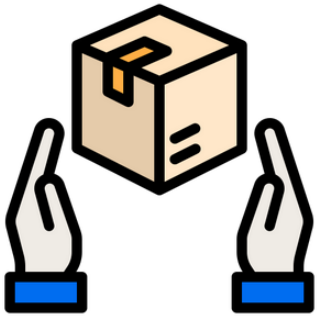
```css
a {
    transition: color 0.3s, background-color 0.3s;
}

a:hover {
    color: white;
    background-color: black;
}
```

Pongamos en práctica lo visto

Comenzaremos creando una tarjeta de producto con algunas pseudoclases aplicadas:

```html
<div class="product-card">
    <img src="producto.png" alt="Producto" width="150">
    <h3>Nombre del Producto</h3>
    <p class="price">19.99</p>
    <button>Añadir al carrito</button>
</div>
```

Sin aplicar nada de css se tendría que ver algo así



Nombre del Producto

19.99

Añadir al carrito

Definamos unos cuantos estilos y apliquemos pseudoclases:

```css
.product-card {
    border: 1px solid #ccc;
    padding: 20px;
    text-align: center;
    width: 200px;
}
.product-card button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
}
.product-card button:hover {
    background-color: #45A049;
}
```

```css
.product-card .price {
    color: #999;
    font-size: 18px;
}
.product-card h3 {
    position: relative;
    display: inline-block;
}
.product-card h3:hover {
    cursor: crosshair;
}
```

Con estos pocos estilos ya tenemos una card y con algunas pseudoclases resulta algo animada, agreguemos pseudolementos ahora

```css
.product-card .price::before {
    content: "$";
}
.product-card h3::after {
    content: '';
    position: absolute;
    left: 0;
    bottom: -5px;
    width: 0;
    height: 2px;
    background-color: #4CAF50;
    transition: width 0.3s ease;
}
.product-card h3:hover::after {
    width: 100%;
}
```

Estamos creando pseudoelementos ::after debajo del título, con un ancho inicial de 0 y una transición suave. Cuando se pasa el cursor sobre el título, el pseudoelemento se anima para tener un ancho del 100%, creando un efecto de subrayado.

Finalmente, agregaremos una animación con @keyframes al elemento de imagen para que gire suavemente al cargar la página:

```css
@keyframes spin {
    0% {
        transform: rotate(0deg);
    }

    100% {
        transform: rotate(360deg);
    }
}

.product-card img {
    width: 200px;
    height: 200px;
    animation: spin 2s ease-in-out;
}
```

Definimos una animación llamada spin que gira el elemento 360 grados.
Luego, aplicamos esta animación a la imagen del producto con una duración de 2 segundos y una curva de aceleración ease-in-out.

Resultado final

- Al combinar todo el código CSS con el HTML, obtendremos una tarjeta de producto con:

- Un botón que cambia de color al pasar el cursor.

- Un precio con el símbolo de dólar añadido con un pseudoelemento.

- Un título con un efecto de subrayado animado al pasar el cursor, gracias a un pseudoelemento y una transición.

- Una imagen que gira suavemente al cargar la página, gracias a una animación con @keyframes.

Este mini proyecto demuestra cómo utilizar pseudoclases, pseudoelementos y animaciones CSS para crear efectos visuales atractivos y mejorar la experiencia del usuario en un sitio web.