07 - CSS Avanzado

Posicionamiento en CSS

El posicionamiento es una parte fundamental para crear diseños complejos y colocar los elementos exactamente donde los necesitamos en la página. CSS ofrece varias formas de posicionar los elementos.

Propiedad position

La propiedad position tiene varios valores que controlan cómo se colocan los elementos en la página:

Static (por defecto): Los elementos se posicionan en el flujo natural de la página.

```
div {
  position: static;
}
```

Relative: El elemento se posiciona relativo a su posición normal. Se pueden usar propiedades como top, left, right y bottom para moverlo.

```
div {
  position: relative;
  top: 20px;
  left: 30px;
}
```

En este caso, el div se desplazará 20 píxeles hacia abajo y 30 píxeles hacia la derecha desde su posición original.

Absolute: El elemento se posiciona en relación a su contenedor más cercano con position: relative o absolute. Si no hay ningún contenedor relativo, se posiciona en relación al documento entero.

```
div {
  position: absolute;
  top: 50px;
  right: 10px;
}
```

Aquí, el div se colocará 50 píxeles desde la parte superior y 10 píxeles desde la derecha de su contenedor padre.

Fixed: El elemento se posiciona relativo a la ventana del navegador, por lo que no se desplaza al hacer scroll.

```
div {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
}
```

Este estilo es ideal para crear barras de navegación fijas que permanecen en la parte superior de la pantalla mientras el usuario navega.

Sticky: Combina aspectos de relative y fixed. Un elemento con position: sticky se comporta como un elemento relativamente posicionado hasta que el usuario hace scroll, momento en el que se comporta como un elemento fijo.

```
div {
     position: sticky;
     top: 0;
```

```
    }
```

Este div se quedará "pegado" en la parte superior de la ventana al hacer scroll.

Diseño con Flexbox

El modelo de caja flexible o Flexbox es una de las herramientas más poderosas de CSS para organizar y alinear elementos. Permite crear diseños que se adaptan bien a diferentes tamaños de pantalla y facilitan la alineación de elementos sin usar trucos como márgenes o float.

Conceptos básicos de Flexbox

Contenedor Flex: Primero, definimos un contenedor como un contenedor flexible con display: flex;. Todos los elementos hijos se comportarán como "ítems flexibles" dentro del contenedor.

```
.contenedor {
  display: flex;
}
```

Alineación de Elementos: Usamos varias propiedades para alinear los elementos dentro de un contenedor flexible:

- **justify-content:** Alinea los elementos a lo largo del eje principal (horizontal por defecto).
    - *flex-start:* Los ítems se alinean al inicio.
    - *center:* Los ítems se alinean al centro.
    - *space-between:* Coloca espacio entre los ítems.

```
.contenedor {
  display: flex;
  justify-content: center;
}
```

- **align-items:** Alinea los elementos a lo largo del eje secundario (vertical por defecto)
    - *flex-start:* Alinea al inicio.
    - *center:* Alinea en el centro.
    - *stretch:* Estira los elementos para que ocupen todo el alto.

```
.contenedor {
           display: flex;
           align-items: center;
      }
```

Ejemplo de Flexbox

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Ejemplo Flexbox</title>
  <style>
      .contenedor {
      display: flex;
      justify-content: space-between;
      align-items: center;
      background-color: lightblue;
      height: 100vh;
      }

      .item {
      background-color: navy;
      color: white;
      padding: 20px;
      margin: 10px;
      }
  </style>
</head>
```

```
<body>

  <div class="contenedor">
      <div class="item">Elemento 1</div>
      <div class="item">Elemento 2</div>
      <div class="item">Elemento 3</div>
  </div>

</body>
</html>
```

En este ejemplo, tenemos tres elementos que están distribuidos uniformemente en un contenedor flexible. justify-content: space-between asegura que haya espacio entre ellos, mientras que align-items: center los centra verticalmente.

Diseño con Grid

El CSS Grid es otro sistema para crear layouts, aún más poderoso que Flexbox cuando se trata de organizar elementos en dos dimensiones (filas y columnas).

Conceptos Básicos de CSS Grid

Crear un Contenedor de Grid: Usamos display: grid; para definir un contenedor como una cuadrícula.

```
.grid {
  display: grid;
}
```

Definir Columnas y Filas: Con grid-template-columns y grid-template-rows podemos definir cuántas columnas y filas tiene nuestra cuadrícula, así como su tamaño.

```
.grid {
  display: grid;
  grid-template-columns: 200px 200px 200px;
  grid-template-rows: 150px 150px;
```

```
}
```

En este ejemplo, la cuadrícula tiene tres columnas de 200px de ancho y dos filas de 150px de alto.

Ajuste Automático de Columnas y Filas: Podemos usar la función repeat y valores como auto o 1fr (fracción) para hacer que las columnas y filas se ajusten automáticamente.

```
.grid {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      grid-template-rows: auto;
      }
```

Aquí, las columnas tendrán tamaños iguales y se ajustarán al tamaño disponible, mientras que las filas se ajustarán automáticamente a la altura del contenido.

Ejemplo de CSS Grid

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Ejemplo CSS Grid</title>
  <style>
      .grid {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 10px;
      background-color: lightgrey;
      padding: 10px;
      }
```

```
      .item {
      background-color: darkblue;
      color: white;
      padding: 20px;
      text-align: center;
      }
  </style>
</head>
<body>

  <div class="grid">
      <div class="item">Item 1</div>
      <div class="item">Item 2</div>
      <div class="item">Item 3</div>
      <div class="item">Item 4</div>
      <div class="item">Item 5</div>
      <div class="item">Item 6</div>
  </div>

</body>
</html>
```

Este ejemplo muestra una cuadrícula de tres columnas, donde cada "item" ocupa una celda y se ajusta perfectamente dentro del espacio disponible.

Responsive

¡Hablemos de diseño responsivo y media queries! Esto es fundamental para que nuestras páginas web se vean bien en cualquier dispositivo, ya sea una computadora de escritorio, tablet o teléfono móvil. Vamos a ver cómo hacerlo de manera sencilla y práctica.

¿Qué es el diseño responsivo?

El diseño responsivo es un enfoque del desarrollo web que hace que las páginas se adapten automáticamente al tamaño de la pantalla del usuario. Así, no importa si alguien está viendo tu página en un celular o en una pantalla gigante, siempre se verá correctamente.

Para lograr esto, usamos media queries, una herramienta en CSS que nos permite aplicar diferentes estilos según el tamaño de la pantalla o las características del dispositivo.

¿Qué son las Media Queries?

Las media queries son reglas de CSS que permiten aplicar diferentes estilos dependiendo del ancho de la pantalla del dispositivo. Por ejemplo, puedes cambiar el diseño de una página cuando la pantalla es menor de 768px de ancho (el tamaño típico de un celular).

Sintaxis de una media query

```
@media (max-width: 768px) {
  /* Estilos aplicados cuando la pantalla es de 768px o más
pequeña */
}
```

Esto significa que los estilos que pongas dentro de esta media query solo se aplicarán si la pantalla es más pequeña o igual a 768px de ancho.

Ejemplo básico de media queries

Vamos a ver un ejemplo sencillo. Supongamos que tienes una página con un título grande y quieres que el tamaño del texto se reduzca en pantallas pequeñas:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Ejemplo de Diseño Responsivo</title>
```

```
  <style>
      h1 {
      font-size: 40px; /* Tamaño grande para pantallas grandes */
      color: blue;
      }

      /* Cambiar el tamaño del texto para pantallas pequeñas */
      @media (max-width: 768px) {
      h1 {
      font-size: 25px; /* Tamaño más pequeño para pantallas móviles
*/
      color: red;
      }
      }
  </style>
</head>
<body>
  <h1>¡Hola, Diseño Responsivo!</h1>
</body>
</html>
```

¿Qué pasa aquí?

En pantallas grandes, el título es azul y tiene un tamaño de 40px.

Cuando la pantalla es de 768px o más pequeña (como un celular), el título se hace más pequeño (25px) y cambia de color a rojo.

Diseño de columnas responsivo

Imagina que tienes dos columnas en una página, y quieres que se vean una al lado de la otra en pantallas grandes, pero una debajo de la otra en pantallas pequeñas. Esto se hace muy fácilmente con media queries.

Ejemplo con dos columnas:

```
<!DOCTYPE html>
<html lang="es">
<head>
```

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Columnas Responsivas</title>
  <style>
      .container {
      display: flex;
      justify-content: space-between;
      }

      .col {
      width: 48%; /* Cada columna ocupa el 48% del ancho en
pantallas grandes */
      padding: 20px;
      background-color: lightgray;
      margin: 10px;
      }

      /* Para pantallas pequeñas, las columnas se apilan */
      @media (max-width: 768px) {
      .container {
      flex-direction: column;
      }
      .col {
      width: 100%; /* Las columnas ocupan el 100% del ancho en
pantallas pequeñas */
      }
      }
  </style>
</head>
<body>
  <div class="container">
      <div class="col">Columna 1</div>
      <div class="col">Columna 2</div>
  </div>
</body>
</html>
```

¿Qué pasa aquí?

En pantallas grandes, las dos columnas están una al lado de la otra y ocupan el 48% del ancho de la página.

En pantallas pequeñas (menos de 768px), las columnas se apilan una debajo de la otra y ocupan el 100% del ancho de la pantalla.

Diseño de imágenes responsivas

Las imágenes también deben adaptarse al tamaño de la pantalla. Para esto, puedes usar una regla simple de CSS:

```
img {
  max-width: 100%;
  height: auto;
}
```

Ejemplo con imágenes responsivas:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Imagen Responsiva</title>
  <style>
      img {
      max-width: 100%;
      height: auto; /* La altura se ajusta automáticamente según el
ancho */
      }
  </style>
</head>
<body>
  <h1>Imagen Responsiva</h1>
  <img src="https://via.placeholder.com/1200x500" alt="Ejemplo de
imagen">
```

```
</body>
</html>
```

¿Qué pasa aquí?

La imagen se ajusta al ancho de la pantalla sin perder su proporción.

Esto es útil cuando la pantalla es pequeña, ya que evita que la imagen se desborde fuera de los bordes de la página.

Media Queries para distintos tamaños de pantalla

Además de la media query básica para pantallas pequeñas, puedes definir varios puntos de quiebre para distintos dispositivos. Aquí algunos ejemplos típicos:

```
/* Pantallas pequeñas (móviles) */
@media (max-width: 576px) {
  /* Estilos para móviles */
}

/* Pantallas medianas (tablets) */
@media (max-width: 768px) {
  /* Estilos para tablets */
}

/* Pantallas grandes (laptops, desktops) */
@media (max-width: 992px) {
  /* Estilos para pantallas grandes */
}

/* Pantallas muy grandes (desktops grandes) */
@media (max-width: 1200px) {
  /* Estilos para pantallas muy grandes */
}
```

Ejercicio práctico: Página de portafolio responsiva

Vamos a crear una página de portafolio que se vea bien tanto en computadoras como en celulares.

HTML básico:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Portafolio Responsivo</title>
  <style>
     /* Estilos generales */
     body {
     font-family: Arial, sans-serif;
     padding: 20px;
     }
     h1 {
     text-align: center;
     margin-bottom: 20px;
     }
     .portfolio {
     display: flex;
     justify-content: space-between;
     }
     .project {
     width: 30%;
     padding: 15px;
     background-color: #f0f0f0;
     margin: 10px;
     text-align: center;
     }

     /* Media query para pantallas pequeñas */
     @media (max-width: 768px) {
     .portfolio {
     flex-direction: column;
     }
```

```
      .project {
      width: 100%;
      }
      }
  </style>
</head>
<body>
  <h1>Mi Portafolio</h1>
  <div class="portfolio">
      <div class="project">Proyecto 1</div>
      <div class="project">Proyecto 2</div>
      <div class="project">Proyecto 3</div>
  </div>
</body>
</html>
```

¿Qué pasa aquí?

En pantallas grandes, los proyectos están alineados en filas, ocupando el 30% del ancho de la página.

En pantallas pequeñas (menos de 768px), los proyectos se apilan verticalmente y ocupan todo el ancho de la pantalla.

¡Y eso es todo! Ahora sabes cómo usar media queries y aplicar diseño responsivo en tu sitio web, lo que hará que tu página se vea increíble en cualquier dispositivo. ¡Practica y experimenta con diferentes puntos de quiebre y verás cómo mejora la experiencia de usuario!