**Guía Media Queries**

<https://www.w3schools.com/css/css3_mediaqueries.asp>

<https://www.geeksforgeeks.org/css-tutorial/?ref=shm>

<https://www.tutorialesprogramacionya.com/cssya/css3ya/index.php>

**¿Qué son las Media Queries?**

Las media queries son una característica de CSS que permite aplicar estilos condicionalmente dependiendo de ciertas características del dispositivo, como:

* Ancho y alto de la ventana del navegador
* Orientación del dispositivo (vertical u horizontal)
* Resolución de la pantalla
* Tipo de dispositivo (pantalla, impresora, etc.)

**Sintaxis básica**

@media tipo-de-medio and (condición) {

/\* Reglas CSS que se aplicarán cuando se cumpla la condición \*/

}

**Tipos de medios comunes:**

* screen: Para pantallas de computadoras, tablets y smartphones
* print: Para documentos impresos y vista previa de impresión
* all: Para todos los dispositivos (valor por defecto)

**Condiciones comunes:**

* width: Ancho exacto de la ventana
* min-width: Ancho mínimo de la ventana
* max-width: Ancho máximo de la ventana
* height: Alto exacto de la ventana
* min-height: Alto mínimo de la ventana
* max-height: Alto máximo de la ventana
* orientation: Orientación del dispositivo (portrait o landscape)
* aspect-ratio: Relación de aspecto de la ventana
* resolution: Resolución del dispositivo

**Ejemplos prácticos**

1. **Aplicar estilos cuando el ancho es menor a 768px** (típico para tablets y smartphones):

@media screen and (max-width: 768px) {

    /\* Estilos para dispositivos con ancho máximo de 768px \*/

  }

1. **Aplicar estilos cuando el ancho está entre 768px y 1024px** (típico para tablets):

@media screen and (min-width: 768px) and (max-width: 1024px) {

    /\* Estilos para dispositivos con ancho entre 768px y 1024px \*/

  }

1. **Aplicar estilos cuando el dispositivo está en orientación horizontal**:

@media screen and (orientation: landscape) {

    /\* Estilos para dispositivos en orientación horizontal \*/

  }

1. **Aplicar estilos para impresión**:

@media print {

    /\* Estilos que solo se aplicarán al imprimir \*/

    .no-print {

      display: none;

    }

  }

**Dimensiones en CSS**

En CSS, puedes especificar dimensiones utilizando diferentes unidades de medida:

**Unidades absolutas:**

* px (píxeles): Unidad fija, 1px = 1 punto en la pantalla
* cm (centímetros): Unidad física
* mm (milímetros): Unidad física
* in (pulgadas): 1in = 96px
* pt (puntos): 1pt = 1/72 de pulgada
* pc (picas): 1pc = 12pt

**Unidades relativas (más recomendadas para diseños responsivos):**

* % (porcentaje): Relativo al elemento padre
* em: Relativo al tamaño de fuente del elemento padre
* rem: Relativo al tamaño de fuente del elemento raíz (generalmente <html>)
* vw: 1% del ancho de la ventana
* vh: 1% del alto de la ventana
* vmin: 1% de la dimensión más pequeña (ancho o alto)
* vmax: 1% de la dimensión más grande (ancho o alto)

**Mejores prácticas para Media Queries**

1. **Mobile-first**: Comienza con estilos para dispositivos móviles y luego agrega media queries para pantallas más grandes.

/\* Estilos base para móviles \*/

.elemento {

    width: 100%;

  }

  /\* Estilos para tablets y escritorio \*/

  @media screen and (min-width: 768px) {

    .elemento {

      width: 50%;

    }

  }

1. **Puntos de quiebre comunes** (breakpoints):
   * 320px - 480px: Smartphones
   * 481px - 768px: Tablets pequeñas
   * 769px - 1024px: Tablets grandes y laptops pequeñas
   * 1025px - 1200px: Laptops y escritorios
   * 1201px y más: Pantallas grandes
2. **Utiliza unidades relativas** como %, em, rem, vw, vh para crear diseños más flexibles.
3. **Prueba en dispositivos reales** o utiliza las herramientas de desarrollo del navegador para simular diferentes tamaños de pantalla.
4. **Considera la densidad de píxeles** para imágenes y elementos visuales:

@media screen and (min-resolution: 2dppx) {

    /\* Estilos para pantallas de alta densidad (Retina) \*/

  }

**Viewport y Meta Tag**

El meta tag viewport es crucial para el funcionamiento correcto de las media queries en dispositivos móviles:

html

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Este tag:

* width=device-width: Establece el ancho de la página al ancho del dispositivo
* initial-scale=1.0: Establece el nivel de zoom inicial a 1 (sin zoom)

Sin este meta tag, los sitios web aparecerían muy pequeños en dispositivos móviles, ya que el navegador intentaría mostrar el sitio completo en la pantalla pequeña.

**Uso de Media Queries**

/\* Estilos base para todos los dispositivos \*/

body {

    font-size: 16px;

    line-height: 1.6;

  }

  .container {

    width: 100%;

    padding: 0 15px;

  }

  /\* Smartphones (orientación vertical) \*/

  @media screen and (max-width: 480px) {

    body {

      font-size: 14px;

    }

    .container {

      padding: 0 10px;

    }

  }

  /\* Tablets y smartphones (orientación horizontal) \*/

  @media screen and (min-width: 481px) and (max-width: 768px) {

    .container {

      max-width: 720px;

      margin: 0 auto;

    }

  }

  /\* Laptops y tablets grandes \*/

  @media screen and (min-width: 769px) and (max-width: 1024px) {

    .container {

      max-width: 960px;

      margin: 0 auto;

    }

  }

  /\* Escritorios \*/

  @media screen and (min-width: 1025px) {

    .container {

      max-width: 1140px;

      margin: 0 auto;

    }

    body {

      font-size: 18px;

    }

  }

  /\* Impresión \*/

  @media print {

    .no-print {

      display: none;

    }

    body {

      font-size: 12pt;

      line-height: 1.5;

    }

  }

**Guía Completa de Flexbox**

Flexbox (Flexible Box Layout) es un modelo de diseño en CSS diseñado para facilitar la creación de diseños complejos y responsivos. Vamos a explorar todos sus aspectos fundamentales:

<https://www.w3schools.com/css/css3_mediaqueries.asp>

<https://www.geeksforgeeks.org/css-tutorial/?ref=shm>

<https://www.tutorialesprogramacionya.com/cssya/css3ya/index.php>

**Conceptos Básicos de Flexbox**

**Terminología Clave**

1. **Contenedor Flex**: El elemento padre que tiene display: flex o display: inline-flex.
2. **Elementos Flex**: Los hijos directos del contenedor flex.
3. **Eje Principal**: El eje primario a lo largo del cual se colocan los elementos (horizontal o vertical).
4. **Eje Cruzado**: El eje perpendicular al eje principal.
5. **Inicio/Fin Principal**: Los lados de inicio y fin en la dirección del eje principal.
6. **Inicio/Fin Cruzado**: Los lados de inicio y fin en la dirección del eje cruzado.

**Activar Flexbox**

.contenedor {

    display: flex;        /\* Convierte en bloque flex \*/

    /\* o \*/

    display: inline-flex; /\* Convierte en elemento inline flex \*/

  }

**Propiedades del Contenedor Flex**

**Dirección del Flujo**

.contenedor {

    flex-direction: row;            /\* Izquierda a derecha (default) \*/

    /\* o \*/

    flex-direction: row-reverse;    /\* Derecha a izquierda \*/

    /\* o \*/

    flex-direction: column;         /\* Arriba hacia abajo \*/

    /\* o \*/

    flex-direction: column-reverse; /\* Abajo hacia arriba \*/

  }

**Envolver Elementos**

.contenedor {

    flex-wrap: nowrap;       /\* No envuelve elementos (default) \*/

    /\* o \*/

    flex-wrap: wrap;         /\* Envuelve elementos si no caben \*/

    /\* o \*/

    flex-wrap: wrap-reverse; /\* Envuelve pero en orden inverso \*/

  }

**Atajo para Dirección y Envoltura**

.contenedor {

    flex-flow: row wrap; /\* Combinación de flex-direction y flex-wrap \*/

  }

**Alineación en el Eje Principal**

.contenedor {

    justify-content: flex-start;    /\* Elementos al inicio (default) \*/

    /\* o \*/

    justify-content: flex-end;      /\* Elementos al final \*/

    /\* o \*/

    justify-content: center;        /\* Elementos centrados \*/

    /\* o \*/

    justify-content: space-between; /\* Espacio entre elementos, pegados a los extremos \*/

    /\* o \*/

    justify-content: space-around;  /\* Espacio alrededor de los elementos \*/

    /\* o \*/

    justify-content: space-evenly;  /\* Espacio equidistante entre elementos \*/

  }

**Alineación en el Eje Cruzado**

.contenedor {

    align-items: stretch;     /\* Estiran para llenar contenedor (default) \*/

    /\* o \*/

    align-items: flex-start;  /\* Elementos al inicio del eje cruzado \*/

    /\* o \*/

    align-items: flex-end;    /\* Elementos al final del eje cruzado \*/

    /\* o \*/

    align-items: center;      /\* Elementos centrados en el eje cruzado \*/

    /\* o \*/

    align-items: baseline;    /\* Alineados por la línea base del texto \*/

  }

**Alineación de Múltiples Líneas**

.contenedor {

    align-content: flex-start;    /\* Líneas agrupadas al inicio \*/

    /\* o \*/

    align-content: flex-end;      /\* Líneas agrupadas al final \*/

    /\* o \*/

    align-content: center;        /\* Líneas centradas \*/

    /\* o \*/

    align-content: space-between; /\* Espacio entre líneas \*/

    /\* o \*/

    align-content: space-around;  /\* Espacio alrededor de líneas \*/

    /\* o \*/

    align-content: stretch;       /\* Líneas estiradas para llenar (default) \*/

  }

**Espaciado entre Elementos**

.contenedor {

    gap: 10px;               /\* Espacio uniforme entre elementos \*/

    /\* o \*/

    gap: 10px 20px;          /\* Espacio entre filas (10px) y columnas (20px) \*/

    /\* o \*/

    row-gap: 10px;           /\* Solo espacio entre filas \*/

    column-gap: 20px;        /\* Solo espacio entre columnas \*/

  }

**Propiedades de los Elementos Flex**

**Orden de Visualización**

.elemento {

    order: 0;    /\* Posición por defecto (0) \*/

    /\* o \*/

    order: 1;    /\* Mayor valor = aparece después \*/

    /\* o \*/

    order: -1;   /\* Menor valor = aparece antes \*/

  }

**Capacidad de Crecimiento**

.elemento {

    flex-grow: 0;    /\* No crece (default) \*/

    /\* o \*/

    flex-grow: 1;    /\* Crece para ocupar espacio disponible \*/

    /\* o \*/

    flex-grow: 2;    /\* Crece al doble de velocidad que flex-grow: 1 \*/

  }

**Capacidad de Reducción**

.elemento {

    flex-shrink: 1;    /\* Puede reducirse si es necesario (default) \*/

    /\* o \*/

    flex-shrink: 0;    /\* No se reduce \*/

    /\* o \*/

    flex-shrink: 2;    /\* Se reduce el doble de rápido que flex-shrink: 1 \*/

  }

**Tamaño Base**

.elemento {

    flex-basis: auto;      /\* Tamaño basado en contenido (default) \*/

    /\* o \*/

    flex-basis: 0;         /\* Tamaño base cero (útil con flex-grow) \*/

    /\* o \*/

    flex-basis: 200px;     /\* Tamaño inicial específico \*/

    /\* o \*/

    flex-basis: 25%;       /\* Tamaño inicial en porcentaje \*/

  }

**Atajo para Grow, Shrink y Basis**

.elemento {

    flex: 0 1 auto;    /\* Default (no crece, puede reducirse, tamaño basado en contenido) \*/

    /\* o \*/

    flex: 1 1 auto;    /\* Puede crecer y reducirse, tamaño basado en contenido \*/

    /\* o \*/

    flex: 1;           /\* Equivalente a flex: 1 1 0% \*/

    /\* o \*/

    flex: auto;        /\* Equivalente a flex: 1 1 auto \*/

    /\* o \*/

    flex: none;        /\* Equivalente a flex: 0 0 auto (no flexible) \*/

  }

**Alineación Individual**

.elemento {

    align-self: auto;        /\* Hereda valor del contenedor (default) \*/

    /\* o \*/

    align-self: flex-start;  /\* Alineado al inicio \*/

    /\* o \*/

    align-self: flex-end;    /\* Alineado al final \*/

    /\* o \*/

    align-self: center;      /\* Centrado \*/

    /\* o \*/

    align-self: stretch;     /\* Estirado \*/

    /\* o \*/

    align-self: baseline;    /\* Alineado con la línea base \*/

  }

**Casos de Uso Comunes**

**Centrado Perfecto (vertical y horizontal)**

.contenedor {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh; /\* Altura completa de la ventana \*/

  }

**Navegación Horizontal**

nav {

    display: flex;

  }

  nav a {

    flex: 0 0 auto; /\* No crece, no se reduce \*/

    padding: 10px 20px;

  }

  /\* Responsive: cambia a vertical en móviles \*/

  @media (max-width: 768px) {

    nav {

      flex-direction: column;

    }

  }

**Diseño con Barra Lateral**

.layout {

    display: flex;

  }

  .sidebar {

    flex: 0 0 250px; /\* Ancho fijo \*/

  }

  .content {

    flex: 1; /\* Ocupa el resto del espacio \*/

  }

  /\* Responsive: apila verticalmente en móviles \*/

  @media (max-width: 768px) {

    .layout {

      flex-direction: column;

    }

    .sidebar {

      flex: none;

      width: 100%;

    }

  }

**Grid de Elementos Iguales**

.grid {

    display: flex;

    flex-wrap: wrap;

    gap: 20px;

  }

  .item {

    flex: 1 1 300px; /\* Crece, se reduce, base 300px \*/

  }

**Holy Grail Layout (3 columnas con cabecera y pie)**

.layout {

    display: flex;

    flex-direction: column;

    min-height: 100vh;

  }

  header, footer {

    flex: 0 0 auto;

  }

  .middle {

    display: flex;

    flex: 1;

  }

  .content {

    flex: 1;

  }

  .left-sidebar, .right-sidebar {

    flex: 0 0 200px;

  }

  /\* Responsive \*/

  @media (max-width: 768px) {

    .middle {

      flex-direction: column;

    }

    .left-sidebar, .right-sidebar {

      flex: none;

    }

  }

**Consideraciones Importantes**

1. **Compatibilidad**: Flexbox es compatible con todos los navegadores modernos, pero pueden existir problemas en versiones muy antiguas.
2. **Anidamiento**: Puedes anidar contenedores flex para crear layouts complejos.
3. **Performance**: Flexbox es muy eficiente para la mayoría de los casos de uso.
4. **Dimensiones**: Flexbox puede ignorar el width y height explícitos según las propiedades flex aplicadas.
5. **Dirección del texto**: Las propiedades de Flexbox pueden comportarse diferente en idiomas con dirección de derecha a izquierda (RTL).
6. **Combinación con Grid**: Para layouts muy complejos, considera combinar Flexbox (para alineación de componentes) con CSS Grid (para estructura de página).