Diseño Interfaces Web

Media Queries

Autores: Victor Manuel y Barlomiej Chudy
Profesor: Gema Rodríguez Flores
Fecha Entrega: 27/01/2026

Autores: Victor Manuel y Barlomiej Chudy
Profesor: Gema Rodríguez Flores

1. ¿Qué son las Media-Queries?

Las ***Media-Queries*** son una funcionalidad de CSS que permite aplicar estilos de forma condicional, es decir, dependiendo de ciertas características del dispositivo o del entorno donde se visualiza la página web.

Mediante ***Media-Queries***, una misma página puede mostrar diferentes estilos según factores como:

- **Adaptar el diseño a diferentes tamaños de pantalla.**
- **Modificar tipografías, márgenes o distribución de elementos.**
- **Optimizar la visualización en móviles y tablets.**
- **Crear interfaces más usables en cualquier dispositivo.**

Su objetivo principal es adaptar el diseño para que el contenido se vea correctamente en diferentes tamaños de pantalla.

Autores: Victor Manuel y Barlomiej Chudy
Profesor: Gema Rodríguez Flores

2. Sintaxis básica

Las Media-Queries se escriben en CSS con la regla ***'@media'***.
Su función es indicar que ciertas reglas de estilo sólo se aplicarán cuando se cumple una condición determinada.

La estructura general es la siguiente:

```
@media (condición) {
  /* Estilos que se aplican solo si la condición se cumple */
}
```

Explicación de la estructura:

- **'@media': Indica que se va a crear una regla condicional.**
- **'(condición)': Es el requisito que debe cumplirse.**
- **'{ }': Dentro del bloque se colocan las reglas CSS que se activarán cuando la condición sea verdadera.**

3. Combinando condiciones

En CSS, las Media Queries no solo permiten aplicar estilos usando una única condición, sino que también permiten combinar varias condiciones para que los estilos se activen únicamente en situaciones más específicas.

Esto es muy útil en diseño responsive, porque un mismo dispositivo puede variar en tamaño u orientación, y no siempre basta con definir un único *breakpoint*, para combinar condiciones usamos principalmente:

- **(and): exige que se cumplan varias condiciones al mismo tiempo.**
- **(,): basta con que se cumpla una de las condiciones.**
- **(not): niega la condición(aplica estilos cuando NO se cumple).**

3.1 Uso de *'AND'*

El operador ***'AND'*** se utiliza para que una Media-Query sólo se active cuando todas las condiciones se cumplen simultáneamente.

Ej: aplicar estilos con *'AND'* sólo en un rango de pantalla *(entre 600px y 900px).*

```css
@media (min-width: 600px) and (max-width: 900px) {
  body {
    font-size: 18px;
  }
}
```

Explicación: Este bloque solo se aplicará únicamente cuando el ancho de la pantalla es de ***600 píxeles*** o más y además sea de ***900 píxeles*** o menos.

Ej: ancho + orientación

```css
@media (max-width: 700px) and (orientation: portrait) {
  .menu {
    font-size: 14px;
  }
}
```

Explicación: Los estilos se aplican sólo si la pantalla es de ***700 píxeles*** o menos y el dispositivo está en vertical.

3.2 Uso de *'OR'*

En Media-Queries, el ***'OR'*** se representa con una coma (**,**) esto significa que si se cumple una de las condiciones, los estilos se aplican.

Ej: Usando *'OR'*

```css
@media (max-width: 500px), (orientation: portrait) {
  body {
    background-color: lightyellow;
  }
}
```

Explicación: El estilo se aplicará si se cumple cualquiera de estas dos condiciones la pantalla mide ***500 píxeles*** o menos o el dispositivo está en orientación vertical, con or no hace falta que se cumplan ambas, con una es suficiente.

3.3 Uso de *'NOT'*

La palabra clave ***'NOT'*** sirve para **negar** una Media-Query, es decir, para aplicar esos estilos cuando la condición **NO** se cumpla.

Ej:

```css
@media not all and (min-width: 900px) {
  body {
    color: red;
  }
}
```

Explicación: Este estilo se aplicará cuando la pantalla **NO** sea de ***900 píxeles*** o más.

4. Ejemplos de uso

Las ***Media-Queries*** se utilizan principalmente para adaptar el diseño de una página web según el tamaño de pantalla o características del dispositivo.

4.1 Ajustar el tamaño del texto en pantallas pequeñas

En caso común es reducir ligeramente el tamaño del texto en móviles para que se lea bien sin ocupar demasiado espacio.

```css
body {
  font-size: 16px;
}

@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

Explicación: En pantallas grandes se usa ***16 píxeles*** como tamaño estándar, en pantallas pequeñas (***600 píxeles*** *o menos*) se reduce a **14 píxeles**.

Esto mejora la lectura y evita que se vea todo demasiado grande en el móvil.

4.2 Cambiar un diseño de columnas a filas (*Flexbox*)

En escritorio es habitual ver elementos colocados en horizontal (*por ejemplo, dos columnas*).
En móvil, estos elementos suelen colocarse en vertical para que no queden apretados.

```css
.contenedor {
  display: flex;
  gap: 20px;
}

@media (max-width: 700px) {
  .contenedor {
    flex-direction: column;
  }
}
```

Explicación: En pantallas grandes los elementos se ven en fila (*row por defecto*), en pantallas pequeñas se cambian a columna para poder adaptarse mejor.

Este ejemplo se usa mucho en:

- **Secciones de imagen + texto.**
- **Bloques de contenido.**
- **Cards.**
- **Formularios de dos columnas.**

4.3 Cambiar la cantidad de columnas usando '*CSS Grid*'

CSS grid nos permite hacer diseños en forma de cuadrícula. Con *Media-Queries* podemos reducir el número de columnas en pantallas que sean más pequeñas.

```css
.galeria {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 10px;
}

@media (max-width: 900px) {
  .galeria {
    grid-template-columns: repeat(2, 1fr);
  }
}

@media (max-width: 500px) {
  .galeria {
    grid-template-columns: 1fr;
  }
}
```

Explicación:

- **Escritorio:** 4 columnas.
- **Pantallas medianas (portátiles):** 2 columnas.
- **Móvil:** 1 columna.

Este patrón es muy común en galería de fotos, catálogos de productos, tarjetas de noticias, etc.

4.4 Ocultar elementos en versiones de móvil

En algunos casos se ocultan elementos grandes que no se adaptan bien a las pantallas pequeñas de los teléfonos.

```css
.menu-escritorio {
  display: block;
}

@media (max-width: 600px) {
  .menu-escritorio {
    display: none;
  }
}
```

Explicación: En las versiones para escritorio o portátil se muestra el menú completo, por el contrario y con la idea de mantener un orden y estética en la web,su versión para móvil se oculta para reemplazarlo por un menú desplegable.

4.5 Mostrar un botón solo en móviles

También se puede hacer lo contrario: ocultar un elemento normalmente y mostrarlo solo en el móvil.

```css
.boton-movil {
  display: none;
}

@media (max-width: 600px) {
  .boton-movil {
    display: block;
  }
}
```

Explicación: Esto es útil para añadir opciones rápidas en móvil, por ejemplo:

- **Botón de ‘realizar llamada’.**
- **Botón de ‘WhatsApp’.**
- **Botón de ‘volver arriba’.**

4.6 Adaptar imágenes para que no se salgan de la pantalla

Para lograr un diseño ***responsive***, a veces interesa que las imágenes tengan un tamaño más grande en escritorio, pero que en móviles se adapten al ancho disponible para que no se salgan del contenedor ni se vean cortadas.

```css
.card img {
  width: 350px;
  height: auto;
}

@media (max-width: 600px) {
  .card img {
    width: 100%;
    max-width: 300px;
  }
}
```

Explicación: En las versiones de escritorio la imagen mide **350 píxeles**, y en sus versiones móviles (*hasta **600 píxeles***) se ajusta al ancho disponible sin deformarse, manteniendo un tamaño máximo para que la imagen se vea bien.

4.7 Ajustar el espaciado (*padding y márgenes*) según pantalla

En escritorio se puede usar más espacio porque hay más ancho disponible, pero en móvil conviene reducirlo.

```css
.seccion {
  padding: 40px;
}

@media (max-width: 600px) {
  .seccion {
    padding: 15px;
  }
}
```

Explicación: Esto hace que en la versión móvil no se pierda tanto espacio en márgenes y el contenido tenga más protagonismo.

5. Ejercicio Práctico

El objetivo de este ejercicio es diseñar una galería de **tres tarjetas** que se adapte a los distintos tamaños de una pantalla: en **escritorio** deberán mostrarse en **tres columnas**, en tablet se mostrarán **dos columnas** y en **móvil** una sola, utilizando las propiedades adecuadas de las *Media-Queries* para realizar estos cambios de una forma responsive.

HTML:

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejercicio Media Queries</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

  <h1>Galería Responsive</h1>

  <div class="galeria">
    <div class="card">
      <h2>Card 1</h2>
      <p>Ejemplo de tarjeta adaptable.</p>
    </div>

    <div class="card">
      <h2>Card 2</h2>
      <p>Usaremos Media Queries para cambiar columnas.</p>
    </div>

    <div class="card">
      <h2>Card 3</h2>
      <p>En móvil se verá una debajo de otra.</p>
    </div>
  </div>

</body>
</html>
```

CSS:

```css
body {
  font-family: Arial, sans-serif;
  background: #f2f2f2;
  padding: 20px;
}

h1 {
  text-align: center;
}

.galeria {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 15px;
  max-width: 900px;
  margin: 0 auto;
}

.card {
  background: white;
  padding: 20px;
  border-radius: 10px;
}
```

Solución:

```css
@media (max-width: 900px) {
  .galeria {
    grid-template-columns: repeat(2, 1fr);
  }
}

@media (max-width: 600px) {
  .galeria {
    grid-template-columns: 1fr;
  }
}
```

6. Preguntas test

1. ¿Qué es una Media Query en CSS?

A) Una función de JavaScript para cambiar estilos.
B) Una regla de CSS que aplica estilos según condiciones del dispositivo.✔
C) Un tipo de etiqueta HTML.
D) Una forma de crear animaciones en CSS.

---

2. ¿Cuál es la estructura correcta de una Media Query?

A) media { (max-width: 600px) }.
B) @media { max-width: 600px }.
C) @media (max-width: 600px) { /* estilos */ } .✔
D) @query (max-width: 600px) { /* estilos */ }.

---

3. ¿Qué significa @media (max-width: 600px)?

A) Se aplica cuando la pantalla mide más de 600px.
B) Se aplica cuando la pantalla mide exactamente 600px.
C) Se aplica cuando la pantalla mide 600px o menos. ✔
D) Se aplica solo en tablets.

---

4. ¿Qué operador se usa para combinar dos condiciones dentro de una Media Query?

A) and ✔
B) or
C) then
D) if