Sumario

INTRODUCCIÓN AL CSS – PARTE 2 – DISEÑO WEB RESPONSIVE

1 INTRODUCCIÓN AL DISEÑO WEB RESPONSIVE

Una maquetación sólida y un diseño responsive son pilares de cualquier interfaz web moderna. No se trata solo de "que se vea bien en móvil", sino de construir experiencias consistentes, accesibles y mantenibles que funcionen en una amplia diversidad de contextos de uso. Estas son las principales ventajas:

- **Experiencia de usuario (UX) consistente**
  - **Adaptabilidad a dispositivos:** el contenido se presenta de forma óptima en móviles, tablets y escritorio, maximizando legibilidad, jerarquía visual y facilidad de interacción.
  - **Rendimiento percibido:** diseños fluidos, imágenes adecuadas y disposición coherente reducen fricción y mejoran tiempos de comprensión.
  - **Accesibilidad real:** respeta tamaños táctiles, contraste, foco visible, y preferencias del usuario (modo oscuro, reducción de movimiento).
- **Escalabilidad y mantenimiento**
  - **Código modular:** patrones claros con Grid/Flexbox y media queries por componente facilitan mantenimiento y colaboración.
  - **Menos "hacks":** CSS más limpio y predecible.
  - **Evolución del producto:** admite nuevas secciones o cambios de prioridad sin rehacer el front-end.
- **Eficiencia de desarrollo**
  - **Reutilización de patrones:** componentes responsive minimizan trabajo repetitivo y acortan iteraciones.
  - **Menos regresiones:** breakpoints guiados por el contenido reducen efectos colaterales.
- **Preparación para el futuro**
  - Diversidad de pantallas: interfaces flexibles y resilientes para distintos tamaños y densidades.

- **Nuevos contextos:** reglas basadas en capacidades (viewport, puntero, hover, color-gamut) se adaptan mejor a dispositivos emergentes.

En resumen, una buena maquetación responsive es una inversión estratégica, mejora la experiencia, acelera el desarrollo, fortalece la accesibilidad y favorece resultados medibles en rendimiento y negocio.

2 ¿QUÉ SON LAS MEDIA QUERIES?

Las media queries son una característica de CSS que permite aplicar estilos de forma condicional en función de características del dispositivo o del entorno de visualización (por ejemplo, ancho de pantalla, altura, resolución, preferencia de color, orientación, entre otras).

Son la base del diseño responsive, ya que permiten que el mismo HTML se adapte a distintos tamaños de pantalla.

2.1 Para qué sirven?

Las **Media Queries** nos sirven para adaptar el diseño de un sitio web a diferentes dispositivos como tablets, smartphones, o cualquier tamaño de pantalla de ordenador, tanto de sobremesa como portátil.

También es posible adaptar los estilos en función de la orientación del dispositivo (vertical u horizontal) e incluso tener unos estilos CSS específicos para impresión, entre otras cosas.

En resúmen, se utilizan para adaptar el layout a móviles, tablets y escritorios.

- Ajustar tipografías y espacios según el tamaño disponible.
- Cargar/mostrar imágenes y componentes alternativos según densidad de píxeles u orientación.
- Respetar preferencias del usuario como modo oscuro o reducción de movimiento.

3 SINTAXIS BÁSICA DE UNA MEDIA QUERY.

```
@media (<condición>) {
 /* Reglas CSS que se aplican si la condición es verdadera */
  }
```

3.1 Ejemplo con ancho mínimo (mobile-first)

Ejemplo con ancho mínimo (mobile-first):

```
/* Estilos base (móviles primero) */
.card {
  padding: 12px;
```

```
}

/* A partir de 768px ajustamos el layout */
@media (min-width: 768px) {
  .grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 16px;
  }
}

/* A partir de 1024px más columnas */
@media (min-width: 1024px) {
  .grid {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

4 TIPOS DE MEDIA QUERIES

A día de hoy, los tipos de medios que puedes definir en una Media Query son los siguientes:

1. **@media screen:** para dispositivos con pantalla o monitor. Es el más común, ya que se usa para ordenadores, tablets, móviles, etc.

```
@media screen {
h1{
background-color: red;
}
}
```

2. **@media print:** para dispositivos de impresión o vistas previas de impresión. Puedes aplicar unos estilos específicos a la hora de imprimir una página web. Es útil para ajustar colores y eliminar elementos no necesarios a la hora de imprimir (como menús, cabeceras, etc.) y, por lo tanto, optimizar el uso de tinta.

```
@media print{
body{
background-color: white;
color: black;
}
}
```

3. **@media all:** se aplica a todos los dispositivos sin excepción.

```
@media all{
p{
font-size: 1rem;
}
```

No es obligatorio definir siempre un tipo de medio, es decir, puedes añadir una nueva regla Media Query simplemente con "@media". En este caso, al no

indicar el tipo de medio, tendría el mismo efecto que utilizar "all" y se aplicaría a todos los dispositivos.

También es posible especificar más de un 'media type' separándolos por comas, por ejemplo:

4.1 TIPOS DE CARACTERÍSTICAS DEL MEDIO

Además del tipo de medio o 'media type', a la hora de utilizar Media Queries también es posible especificar ciertas características del dispositivo de manera condicional. Así no solo podemos aplicar estilos en función del tipo de medio, sino que podemos hacerlo en función de cosas como el ancho de pantalla, la orientación del dispositivo o de las preferencias del sistema que tenga configuradas el usuario. Las 'media types' van entre paréntesis, por ejemplo:

I. **width / height: Condicionan por ancho/alto del viewport.**

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

- min-width / max-width: Patrón típico responsive (mobile-first con min-width).

II. **orientation:** Detecta orientación del dispositivo: portrait o landscape.

```
@media (orientation: landscape) {
        .hero { height: 60vh; } }
```

III. **resolution / device-pixel-ratio:** Densidad de píxeles (retina).

```
@media (min-resolution: 2dppx) {
        .logo { background-image: url(logo@2x.png); } }
```

IV. **prefers-color-scheme:** Preferencia de modo claro/oscuro.

```
@media (prefers-color-scheme: dark) {
        body { background: #111; color: #eee; } }
```

V. **prefers-reduced-motion:** Reduce animaciones para accesibilidad.

```
@media (prefers-reduced-motion: reduce) {
        * { transition: none !important; animation: none !important; } }
```

VI. **pointer / hover:** Capacidades del puntero (fino, tosco) y si soporta hover.

```
@media (hover: hover) and (pointer: fine) {
        .menu:hover { background: #f0f0f0; } }
```

VII. **aspect-ratio:** Relación de aspecto del viewport.

```
@media (min-aspect-ratio: 16/9) {
        .video { max-height: 60vh; } }
```

VIII. **color-gamut:** Gama de color del dispositivo (srgb, p3, rec2020).

```
@media (color-gamut: p3) {
        .banner { color: color(display-p3 0.9 0.2 0.2); } }
```

5 OPERADORES LÓGICOS

Es posible hacer combinaciones utilizando operadores lógicos para crear Media Queries más elaboradas.

Operadores lógicos: and, not, only, comas para OR.

6 ESTRATEGIAS A LA HORA DE UTILIZAR MEDIA QUERIES

En la medida de lo posible, sigue la filosofía **mobile-first**. Es decir, escribe los estilos CSS primero para dispositivos móviles y utiliza las Media Queries para escalar a tamaños superiores

- **Mobile-first:** define estilos base para pantallas pequeñas y añade @media (min-width: Xpx) para ampliar. Suele resultar en CSS más simple y predecible.
- **Desktop-first:** estilos base para pantallas grandes y usas (max-width) para reducir. A veces genera overrides más complejos.

7 CÓMO ELEGIR BREAKPOINTS

Los **breakpoints** son los diferentes anchos de pantalla en los que los estilos deben cambiar. Puedes utilizar los que quieras, pero hay una serie de medidas más o menos estandarizadas para los diferentes tipos de dispositivo. En la medida de lo posible, utilízalas y evita crear muchos **breakpoints** a no ser que el diseño lo requiera estrictamente.

8 CONSEJOS

- Piensa en contenido: que los cambios de breakpoint respondan a saltos reales en el diseño.
- Prefiere mobile-first con (min-width) para evitar cascadas complicadas de overrides.
- Agrupa media queries por componente (cuando convenga) para mantener coherencia y facilitar mantenimiento. De esta manera, el código será más limpio y legible

```
h1{
font-size: 1rem;
@media screen and (min-width: 767px){ font-size: 1.5rem; }
```

```
@media screen and (min-width: 1024px){ font-size: 2rem; }

}
```

- Usa variables y un sistema de diseño (tokens) para breakpoints, tamaños y espacios.
- Testea en dispositivos reales y con herramientas de devtools para diferentes tamaños, orientaciones y emulación de features.
- Evita depender de device-width o user-agent; centra las reglas en el viewport y capacidades.
- Documenta las razones de cada breakpoint en el código o en el sistema de diseño.