📚 Web Responsiva Pura: Guía Paso a Paso (*HTML + CSS + JavaScript sin librerías externas*)

> **Asignatura**: Diseño de Interfaces Web (DIW)
> **Ciclo Formativo**: Grado Superior en Desarrollo de Aplicaciones Web
> **Profesor**: Antonio Sierra Grcía **Fecha**: 21 de enero de 2026

🔍 Introducción

Este documento te guiará para construir una página web moderna, totalmente responsiva, con menú hamburguesa, modo oscuro, modal y animaciones suaves —usando solo tecnologías nativas del navegador.

> ✅ **Objetivo**: Entender los fundamentos del diseño web actual sin depender de frameworks como Bootstrap o Tailwind.

📁 Estructura del proyecto

```
mi-web/
├── index.html
├── style.css
└── script.js
```

Empezaremos desde cero y añadiremos funcionalidades de forma incremental.

🧱 Fase 1: Diseño responsivo básico con menú hamburguesa

🌐 Paso 1: Preparar el HTML base

Crea el archivo `index.html` con esta estructura:

HTML

```html
<!-- 🌐 HTML -->
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <!-- 🔑 IMPORTANTE: Esta etiqueta permite que los móviles muestren la web correctamente -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Web Responsiva</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js" defer></script>
</head>
<body>
  <header class="header">
```

```
      <div class="container">
        <h1 class="logo">MiWeb</h1>

        <!-- Botón hamburguesa (solo visible en móvil) -->
        <button class="menu-toggle" aria-label="Abrir menú">
          <span></span>
          <span></span>
          <span></span>
        </button>

        <!-- Navegación principal -->
        <nav class="nav">
          <ul>
            <li><a href="#">Inicio</a></li>
            <li><a href="#">Servicios</a></li>
            <li><a href="#">Contacto</a></li>
          </ul>
        </nav>
      </div>
    </header>

    <main class="main">
      <div class="container">
        <h2>Nuestros Proyectos</h2>
        <div class="cards-grid">
          <div class="card"><h3>Proyecto A</h3></div>
          <div class="card"><h3>Proyecto B</h3></div>
          <div class="card"><h3>Proyecto C</h3></div>
        </div>
      </div>
    </main>

    <footer class="footer">
      <div class="container">
        <p>&copy; 2026 MiWeb</p>
      </div>
    </footer>
  </body>
  </html>
```

> 💡 **Notas importantes**:
>
> - `<meta name="viewport">` es obligatoria para responsividad.
> - Usamos clases semánticas (`header`, `nav`, `main`, `footer`) para mejorar accesibilidad y legibilidad.
> - El botón hamburguesa está compuesto por tres `<span>` que formarán las "líneas".

🎨 Paso 2: Estilos base (mobile-first)

Crea `style.css` y añade:

CSS

```css
/* 🎨 CSS */
/* === RESET BÁSICO === */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box; /* Incluye padding y border en el ancho total */
}

body {
  font-family: system-ui, -apple-system, sans-serif;
  line-height: 1.6;
  color: #333;
  background-color: #f9f9f9;
}

/* === CONTENEDOR CENTRADO === */
.container {
  width: 90%;           /* Ocupa el 90% del ancho */
  max-width: 1200px;    /* Pero no más de 1200px en pantallas grandes */
  margin: 0 auto;       /* Centra horizontalmente */
  padding: 1rem 0;
}

/* === CABECERA === */
.header {
  background-color: white;
  padding: 1rem 0;
  box-shadow: 0 2px 5px rgba(0,0,0,0.1); /* Sombra sutil */
}

.logo {
  font-size: 1.5rem;
  font-weight: bold;
}

/* === BOTÓN HAMBURGUESA (oculto por ahora) === */
.menu-toggle {
  display: none; /* Solo lo mostraremos en móviles */
  background: none;
  border: none;
  cursor: pointer;
  width: 30px;
  height: 30px;
}

.menu-toggle span {
  display: block;
  height: 3px;
  background-color: #333;
  margin: 5px 0;
  border-radius: 2px;
```

```
}

/* === NAVEGACIÓN === */
.nav ul {
  list-style: none;
  display: flex;          /* Horizontal en escritorio */
  gap: 1.5rem;            /* Espacio entre enlaces */
}

.nav a {
  text-decoration: none;
  color: #333;
  font-weight: 500;
  padding: 0.5rem 0;
}

.nav a:hover {
  color: #007bff; /* Azul al pasar el ratón */
}

/* === TARJETAS === */
.cards-grid {
  display: grid;
  /* Clave de la responsividad:
     - minmax(250px, 1fr): cada columna tiene al menos 250px
     - auto-fit: ajusta cuantas columnas quepan
     - 1fr: reparte el espacio restante equitativamente */
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 2rem;
  margin-top: 1.5rem;
}

.card {
  background: white;
  padding: 1.5rem;
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
}

/* === PIE DE PÁGINA === */
.footer {
  background-color: white;
  text-align: center;
  padding: 1.5rem 0;
  margin-top: 3rem;
  box-shadow: 0 -2px 5px rgba(0,0,0,0.05);
}
```

> 🎯 **Concepto clave**: `mobile-first` Diseñamos primero para pantallas pequeñas. Luego, usamos media queries para mejorar la experiencia en pantallas grandes.

🎨 Paso 3: Hacerlo responsive con media queries

Añade al final de `style.css`:

CSS

```css
/* 🎨 CSS */
/* === MÓVIL: Pantallas menores a 768px === */
@media (max-width: 767px) {
  /* Mostrar el botón hamburguesa */
  .menu-toggle {
    display: block;
  }

  /* Ocultar el menú de navegación por defecto */
  .nav {
    position: absolute;
    top: 100%;
    left: 0;
    right: 0;
    background: white;
    padding: 1rem;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    display: none; /* ¡invisible hasta que se abra! */
  }

  /* Apilar los enlaces en vertical */
  .nav ul {
    flex-direction: column;
    gap: 1rem;
  }

  /* Tarjetas: una sola columna en móvil */
  .cards-grid {
    grid-template-columns: 1fr; /* Solo una columna */
  }
}
```

> 💡 ¿**Por qué** `max-width: 767px`? Es un punto de quiebre común: 768px es el ancho típico de una tablet en vertical. Todo por debajo se considera "móvil".

⚡ Paso 4: Añadir interactividad con JavaScript

Crea `script.js`:

JS

```
// ⚡ JavaScript
// Seleccionamos los elementos del DOM
const menuToggle = document.querySelector('.menu-toggle');
const nav = document.querySelector('.nav');

// Escuchamos el clic en el botón hamburguesa
menuToggle.addEventListener('click', () => {
  // Alternamos la clase "open" → esto activará el CSS
  nav.classList.toggle('open');
});
```

Y en `style.css`, dentro de la media query (`@media (max-width: 767px)`), añade:

CSS

```
/* 🎨 CSS */
/* Cuando el menú está abierto */
.nav.open {
  display: block;
}
```

> ✅ **Resultado**: Al hacer clic en el botón, se añade/elimina la clase open, y el menú aparece o desaparece.

> 🧠 **Buena práctica**: JavaScript solo cambia clases. CSS define cómo se ve. Así separamos lógica de presentación.

✅ **¡Fase 1 completada!** Ya tienes una web totalmente responsiva con:

- Diseño mobile-first.
- Menú hamburguesa funcional.
- Grid de tarjetas adaptable.

🌓 Fase 2: Añadir modo oscuro, modal y animaciones suaves

Ahora mejoraremos la experiencia del usuario con tres nuevas funcionalidades.

🌙 Paso 1: Añadir botón de modo oscuro

En `index.html`, dentro del `<header>` (justo antes de cerrar `</div>` del contenedor):

HTML

```
<!-- 🌐 HTML -->
<button class="theme-toggle" aria-label="Cambiar tema">🌙</button>
```

> 🎨 Puedes usar íconos SVG si quieres, pero el emoji es suficiente para este ejemplo.

🎨 Paso 2: Usar variables CSS para los temas

En `style.css`, arriba del todo (justo después del `*` reset):

CSS

```
/* 🎨 CSS */
/* === VARIABLES DE COLOR === */
:root {
  /* Modo claro (por defecto) */
  --bg-color: #f9f9f9;
  --text-color: #333;
  --card-bg: white;
  --header-bg: white;
  --shadow: rgba(0,0,0,0.1);
}

/* Modo oscuro */
.dark-mode {
  --bg-color: #121212;
  --text-color: #e0e0e0;
  --card-bg: #1e1e1e;
  --header-bg: #1a1a1a;
  --shadow: rgba(0,0,0,0.4);
}
```

Luego, reemplaza los colores fijos por variables:

CSS

```
/* 🎨 CSS */
body {
  background-color: var(--bg-color);
  color: var(--text-color);
  /* Transición suave al cambiar tema */
  transition: background-color 0.3s, color 0.3s;
}
```

```css
.header {
  background-color: var(--header-bg);
  box-shadow: 0 2px 5px var(--shadow);
}

.card {
  background: var(--card-bg);
  box-shadow: 0 2px 8px var(--shadow);
}

.footer {
  background-color: var(--header-bg);
  box-shadow: 0 -2px 5px var(--shadow);
}
```

> 🔑 **Ventaja**: Cambiar el tema es tan fácil como añadir la clase `dark-mode` al `<html>`.

⚡ Paso 3: JavaScript para cambiar y guardar el tema

En `script.js`, añade al final:

JS

```javascript
// ⚡ JavaScript
// === MODO OSCURO ===
const themeToggle = document.querySelector('.theme-toggle');
const htmlElement = document.documentElement; // <html>

// Recuperar tema guardado (si existe)
const savedTheme = localStorage.getItem('theme') || 'light-mode';
htmlElement.className = savedTheme;

// Actualizar ícono del botón
themeToggle.textContent = savedTheme === 'light-mode' ? '🌙' : '🌞';

// Cambiar tema al hacer clic
themeToggle.addEventListener('click', () => {
  const currentTheme = htmlElement.className;
  const newTheme = currentTheme === 'light-mode' ? 'dark-mode' : 'light-mode';

  htmlElement.className = newTheme;
  localStorage.setItem('theme', newTheme); // Guardar en el navegador

  // Actualizar ícono
  themeToggle.textContent = newTheme === 'light-mode' ? '🌙' : '🌞';
});
```

> 💾 **localStorage**: guarda datos en el navegador del usuario. Persiste aunque cierre la pestaña.

Paso 4: Crear un modal simple

Añade antes de `</body>` en `index.html`:

HTML

```
<!-- 🌐 HTML -->
<!-- Modal -->
<div class="modal" id="modal">
  <div class="modal-content">
    <span class="modal-close" id="close-modal">&times;</span>
    <h2>Detalles del Proyecto</h2>
    <p>Este es un cuadro de diálogo creado con tecnologías nativas.</p>
  </div>
</div>
```

Y modifica un enlace del menú para abrirlo (por ejemplo, "Servicios"):

HTML

```
<!-- 🌐 HTML -->
<li><a href="#" id="open-modal">Servicios</a></li>
```

Añade al final de `style.css`:

CSS

```
/* 🎨 CSS */
/* === MODAL === */
.modal {
  display: none;                /* Oculto por defecto */
  position: fixed;              /* Fijo respecto a la ventana */
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0,0,0,0.6); /* Fondo oscuro semitransparente */
  z-index: 1000;                /* Aparece encima de todo */
}
```

```css
.modal-content {
  background: var(--card-bg);
  margin: 10% auto;             /* Centrado vertical aproximado */
  padding: 2rem;
  width: 90%;
  max-width: 500px;
  border-radius: 10px;
  box-shadow: 0 10px 30px var(--shadow);
  position: relative;
}

.modal-close {
  position: absolute;
  top: 1rem;
  right: 1.5rem;
  font-size: 1.8rem;
  cursor: pointer;
  color: var(--text-color);
}

.modal-close:hover {
  color: #ff4d4d;
}
```

En `script.js` añade:

JS

```js
// ⚡ JavaScript
// === MODAL ===
const modal = document.getElementById('modal');
const openModalBtn = document.getElementById('open-modal');
const closeModalBtn = document.getElementById('close-modal');

// Abrir modal
openModalBtn.addEventListener('click', (e) => {
  e.preventDefault(); // Evita que el enlace recargue la página
  modal.style.display = 'block';
});

// Cerrar modal (con la X)
closeModalBtn.addEventListener('click', () => {
  modal.style.display = 'none';
});

// Cerrar modal al hacer clic fuera del contenido
window.addEventListener('click', (e) => {
  if (e.target === modal) {
    modal.style.display = 'none';
```

```
  }
});
```

> ✅ **Accesibilidad**: El modal debe poder cerrarse con teclado (esto se puede mejorar con keydown, pero ya es un buen inicio).

🎥 Paso 5: Añadir animaciones suaves

Transiciones en tarjetas

En `.card` (en `style.css`):

CSS

```css
/* 🎨 CSS */
.card {
  /* ... otros estilos ... */
  transition: transform 0.3s, box-shadow 0.3s; /* Animación al interactuar */
}

.card:hover {
  transform: translateY(-5px); /* Sube ligeramente */
  box-shadow: 0 6px 20px var(--shadow); /* Sombra más intensa */
}
```

Animaciones al abrir el modal (opcional pero elegante)

Añade estas animaciones en `style.css`:

CSS

```css
/* 🎨 CSS */
/* Animación de fundido */
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}

/* Animación de deslizamiento */
@keyframes slideIn {
  from {
    opacity: 0;
    transform: translateY(-20px);
  }
```

```
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

/* Aplicar animaciones */
.modal {
  animation: fadeIn 0.3s ease;
}

.modal-content {
  animation: slideIn 0.4s ease;
}
```

> ⏱️ **Duración ideal**: entre 0.2s y 0.4s. Las animaciones lentas molestan; las rápidas dan sensación de fluidez.

📌 Resumen de aprendizajes

Concepto	Tecnología usada	¿Por qué es importante?
Responsividad	`viewport`, media queries, Grid	La web debe verse bien en cualquier dispositivo
Menú hamburguesa	`classList.toggle()`, posición absoluta	Patrón estándar en móviles
Modo oscuro	Variables CSS, `localStorage`	Mejora la accesibilidad y experiencia de usuario
Modal	`position: fixed`, eventos de clic	Patrón común para mostrar información contextual
Animaciones	`transition`, `@keyframes`	Interfaces más agradables y profesionales

🛠️ Consejos finales

1. Empieza siempre con mobile-first.
2. Usa variables CSS para colores, espaciados y tamaños.
3. Evita JavaScript para diseño: deja que CSS maneje la apariencia.
4. Prueba en dispositivos reales, no solo en el navegador.
5. Sé minimalista: menos código = menos errores.