Diplomatura en

Programación FullStack Developer

GLOBAL ACADEMY

CSS
Responsive Design y Media Queries
Adaptando interfaces para todos los dispositivos

Responsive Design

Definición:
El diseño responsivo es un enfoque de diseño web que busca asegurar que las páginas web funcionen bien en una variedad de dispositivos y tamaños de ventana o pantalla. Se logra mediante el uso de fluid grid layouts, imágenes flexibles y media queries.

Meta:
Asegurar una buena experiencia de usuario en cualquier dispositivo, desde teléfonos móviles hasta pantallas grandes.

Importancia del Diseño Responsivo

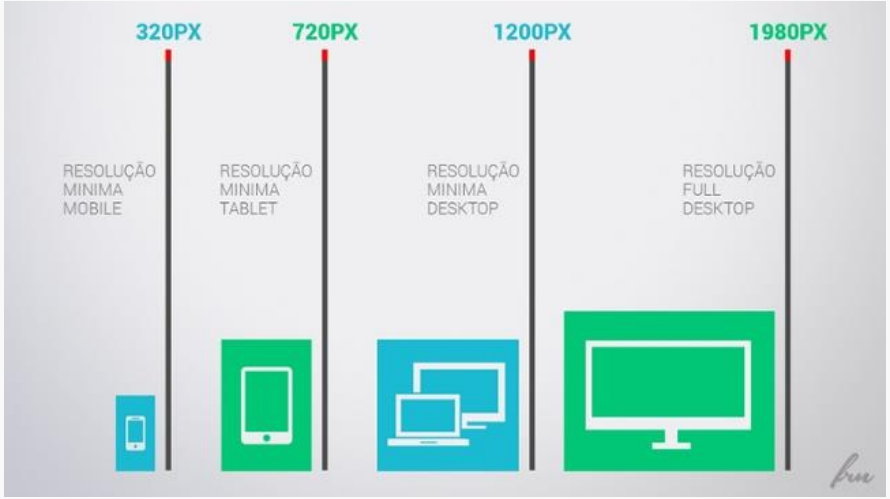
Accesibilidad Universal:
En un mundo donde el acceso a internet se realiza cada vez más desde dispositivos móviles, el diseño responsivo permite que todos los usuarios tengan una experiencia óptima.

SEO y Visibilidad en la Web:
Google y otros motores de búsqueda priorizan en sus resultados a sitios web responsivos, lo que es crucial para la visibilidad en línea.

Breve Historia del Diseño Responsivo

Orígenes:
Aunque la práctica de diseño adaptable comenzó a ganar atención en los años 2000, fue Ethan Marcotte quien acuñó el término "Responsive Web Design" en 2010.

Evolución:
Desde su concepción, el diseño responsivo ha evolucionado de simples layouts fluidos a complejas interfaces que responden no solo al tamaño de pantalla sino también a las preferencias del usuario.



Flujo de Contenido Fluido

Concepto: El contenido debe fluir libremente en el espacio disponible, adaptándose al tamaño de la pantalla sin romper el diseño.

Cómo lograrlo: Utilizar layouts basados en porcentajes y evitar medidas fijas en píxeles que puedan restringir la fluidez del diseño

Unidades Relativas

Importancia: Las unidades relativas se adaptan mejor a diferentes resoluciones de pantalla en comparación con las unidades absolutas (como píxeles).

Ejemplos de unidades relativas:

- **Porcentajes (%):** Se adapta al tamaño del contenedor.
- **Em y Rem:** Se basan en el tamaño de fuente del elemento raíz, ideal para tipos de letra y espacios entre elementos.
- **Viewport Width (VW) y Viewport Height (VH):** Proporciones del ancho o alto de la ventana visible del navegador, útiles para elementos que deben ajustarse al tamaño de la pantalla.

Imágenes Flexibles

Problema: Las imágenes de tamaño fijo pueden desbordar sus contenedores o ser demasiado pequeñas en dispositivos grandes.

Solución:

Uso de max-width: 100% para asegurar que las imágenes nunca sean más anchas que su contenedor.

Aplicar funciones de tamaño en CSS como object-fit para controlar cómo se debe llenar el espacio del contenedor sin deformar la imagen.

Ejemplo:

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Diseño Responsivo Mejorado</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="contenedor">
        <p>Este texto se adapta al tamaño del contenedor gracias al uso de unidades relativas.</p>
        <div class="contenedor-imagen">
            <img src="imagen-responsiva.jpg" alt="Imagen responsiva">
        </div>
    </div>
</body>
```

```css
body {
    font-size: 16px; /* Tamaño de fuente base */
}
.contenedor {
    width: 80%; /* Ancho relativo al tamaño de la ventana */
    margin: auto; /* Centrar el contenedor */
    border: 2px solid #000;
    /* Bordes para visualizar el contenedor */
    padding: 20px;
}
.contenedor p {
    font-size: 1.25rem;
    /* Tamaño de fuente relativo al tamaño de fuente base del body */
}
.contenedor-imagen {
    width: 100%; /* Ancho completo dentro del contenedor */
    height: 300px; /* Altura fija para demostrar object-fit */
    overflow: hidden; /* Evita que la imagen desborde el contenedor */
}
img {
    width: 100%;
    height: 100%;
    object-fit: cover;
    /* Asegura que la imagen cubra completamente el área sin distorsionarse */
    display: block;
}
```

Contenedor de Imagen: Establece un área específica con una altura fija de 300px, permitiendo demostrar efectivamente cómo object-fit puede influir en el comportamiento de la imagen dentro de un espacio delimitado.

Imagen: Aplica width: 100% y height: 100% para llenar completamente el espacio del contenedor de la imagen. La propiedad object-fit: cover asegura que la imagen cubra todo el espacio disponible, recortando el exceso si es necesario, sin distorsionar sus proporciones originales.

La propiedad CSS object-fit permite controlar el ajuste del contenido de un elemento <img> o <video> dentro de su contenedor.
Aquí están los valores que puede tomar:

fill:
Este es el valor predeterminado. El contenido se estira para ajustarse completamente al contenedor, lo que puede resultar en una distorsión si las proporciones del contenido no coinciden con las del contenedor.

contain:
El contenido se escala para mantener su aspecto original mientras se ajusta dentro del contenedor. Todo el contenido es visible, pero esto puede resultar en espacio adicional en el contenedor si las proporciones no coinciden.

cover:
El contenido se cubre completamente en el contenedor manteniendo su aspecto original. Parte del contenido puede ser recortado si las proporciones no coinciden.

none:
El contenido no se escala. Se mostrará su tamaño original, lo que podría llevar a que se desborde del contenedor si es demasiado grande.

scale-down:
Compara los efectos de none y contain para proporcionar el resultado visual más pequeño. Es útil cuando el tamaño original del contenido es más grande que el contenedor, pero queremos mostrar todo el contenido sin recortar y sin distorsión.

CSS Media Queries - Concepto General

¿Qué son las Media Queries?

- **Definición:** Las Media Queries son una técnica en CSS que permite aplicar estilos condicionales basados en características del medio de visualización, como el tamaño de la pantalla, la resolución, la orientación, entre otros.

- **Utilidad:** Permiten que los diseñadores y desarrolladores creen experiencias visuales adaptativas y específicas para diferentes dispositivos y contextos de visualización.

Estructura Básica de una Media Query

```
@media (media-feature) {
    /* Estilos que se aplican si la condición es verdadera */
}
```

Componentes:

- **Tipo de medio:** Por ejemplo, screen, print, all.

- **Media feature:** Características del medio, como width, orientation, resolution.

Ejemplo de Uso Básico de Media Queries

```
body {
    background-color: lightblue;
    /* Color predeterminado */
}

@media (min-width: 600px) {
    body {
        background-color: coral;
        /* Cambia a coral si el ancho es al menos 600px */
    }
}
```

Cambiar el color de fondo de una página según el ancho de la pantalla.

Cambiar el color de fondo de una página según el ancho de la pantalla:

```
body {
    font-size: 16px;
    /* Tamaño predeterminado para escritorio */
}

@media (max-width: 1200px) {
    body {
        font-size: 14px;
        /* Para notebooks */
    }
}

@media (max-width: 768px) {
    body {
        font-size: 12px;
        /* Para tabletas */
    }
}
```

Ajustar el tamaño del texto para que sea legible en todos los dispositivos.

Desktops (>1200px): Uso de un tamaño de fuente estándar adecuado para lectura en pantallas grandes.

Notebooks (<=1200px): Reducción del tamaño de la fuente para mantener la legibilidad y el aprovechamiento del espacio.

Tabletas (<=768px): Ajuste a un tamaño de fuente menor para optimizar la visibilidad y el manejo con una sola mano.

Beneficios de las Media Queries en Responsive Design

- **Flexibilidad:** Adaptación fluida del contenido a cualquier tamaño de pantalla, proporcionando una experiencia de usuario coherente y accesible.

- **Control Detallado:** Permite ajustes precisos para cada rango de dispositivo, optimizando cada vista según las necesidades específicas del usuario y del dispositivo.

Breakpoints en Responsive Design

Definición de Breakpoints

- **¿Qué son los Breakpoints?:** Los breakpoints o puntos de ruptura son condiciones dentro de las Media Queries donde el contenido de un sitio web cambia de manera significativa para adaptarse a diferentes tamaños de pantalla.

- **Propósito:** Asegurar que el diseño del sitio sea óptimo y funcional en diferentes dispositivos, desde móviles hasta desktops.

Determinación de Breakpoints

Basado en el Contenido:

Seleccionar breakpoints donde el contenido naturalmente "rompe" o parece desordenado, en lugar de adherirse a dimensiones específicas de dispositivos.

Ejemplos Comunes:

- Pequeños dispositivos móviles: 320px, 480px
- Grandes dispositivos móviles: 600px
- Tabletas: 768px, 1024px
- Escritorios: 1280px, 1366px, 1920px

Ajuste de un Layout de Columnas:

```
.container {
    display: grid;
    grid-template-columns: 1fr;
    /* Columna única por defecto */
}

@media (min-width: 600px) {
    .container {
        grid-template-columns: repeat(2, 1fr);
        /* Dos columnas para tablets y arriba */
    }
}

@media (min-width: 1024px) {
    .container {
        grid-template-columns: repeat(3, 1fr);
        /* Tres columnas para escritorios */
    }
}
```

Unidades en CSS Relevantes para Responsive Design

Unidades Relativas

Porcentajes (%):

- Descripción: Se calculan en relación con alguna propiedad de su elemento contenedor.
- Ejemplo: width: 80%; hace que un elemento ocupe el 80% del ancho de su contenedor.

Em y Rem:

- Descripción: em se basa en el tamaño de fuente del elemento padre, mientras que rem se refiere al tamaño de fuente del elemento raíz.
- Ejemplo: font-size: 1.5em; o margin: 0.5rem;

Viewport Width (VW) y Viewport Height (VH):

- Descripción: Estas unidades son una porción del ancho o alto de la ventana gráfica.
- Ejemplo: height: 50vh; hace que un elemento tenga la mitad de la altura de la ventana gráfica.

Unidades de Función Mínima, Máxima y de Rango

min(), max(), y clamp():

Descripción:

- Permiten definir un tamaño que se adapta dentro de un rango basado en condiciones específicas.

Ejemplos:

- width: min(100%, 500px); asegura que el ancho no supere los 500px, pero permite que sea menor si el contenedor lo es.
- font-size: clamp(1rem, 2.5vw, 2rem); ajusta el tamaño de fuente dinámicamente entre 1rem y 2rem, basado en el tamaño de la ventana gráfica.

Aplicaciones Prácticas de Estas Unidades

Adaptación Fluida:

Las unidades VW y VH son ideales para layouts que requieren dimensiones que se ajusten de manera fluida con el tamaño de la ventana.

Escalado Proporcional de Texto y Componentes:

Em y Rem son perfectas para ajustar el escalado de componentes y texto de acuerdo al dispositivo sin perder proporcionalidad.

Ejemplo:

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Unidades CSS en Diseño Responsivo</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Bienvenidos</h1>
    </header>
    <section>
        <p>Este párrafo utiliza rem para el tamaño de fuente, adaptable a diferentes dispositivos.</p>
        <div class="responsive-div">
            Este div se adapta al cambiar el tamaño de la ventana, manteniendo su proporcionalidad.
        </div>
    </section>
    <footer>
        <p>Contacto: ejemplo@correo.com</p>
    </footer>
</body>
```

```css
body {
    margin: 0;
    font-family: Arial, sans-serif;
}

header {
    height: 10vh;
    background-color: skyblue;
}

header h1 {
    font-size: clamp(1.5rem, 4vw, 2rem);
    margin: 0;
    padding: 10px;
}

section {
    padding: 2vw;
}

section p {
    font-size: 1.2rem;
}
```

```css
.responsive-div {
    width: max(300px, 50vw);
    height: 20vh;
    background-color: coral;
    padding: 10px;
    box-sizing: border-box;
}

footer {
    height: 5vh;
    background-color: lightgrey;
    padding: 5px;
}

footer p {
    font-size: min(2vw, 1rem);
    margin: 0;
}
```

General (body): Establecimiento del margen a cero y la fuente predeterminada para todo el cuerpo del documento.

Header y Footer: Utilización de unidades vh para determinar la altura relativa a la altura de la ventana de visualización.

Header h1: Uso de clamp() para asegurar que el tamaño del título sea responsive pero esté dentro de un rango legible y adecuado.

Section y p: Se asigna un padding en vw para la sección y se establece un tamaño de fuente en rem para los párrafos, lo que proporciona consistencia y escalabilidad.

.responsive-div: Demostración de cómo max() puede ser utilizado para garantizar que el div no exceda un ancho máximo de 300px, pero pueda ser más pequeño si el 50% del vw es menos de 300px.

Footer p: Uso de min() para seleccionar el menor valor entre dos unidades, garantizando la legibilidad en dispositivos con pantallas pequeñas.

Técnicas para Imágenes Responsivas

Imágenes Flexibles:

- Uso de max-width: 100% y height: auto para asegurar que las imágenes se escalen adecuadamente dentro de sus contenedores sin superar el tamaño original.

Imágenes en CSS Background:

- Utilización de background-size: cover o background-size: contain dependiendo de si se desea que el fondo cubra totalmente el área o se ajuste dentro de ella sin recortes.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Imágenes de Fondo Responsivas</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="background-cover">
        <p>Este fondo utiliza 'cover'</p>
    </div>
    <div class="background-contain">
        <p>Este fondo utiliza 'contain'</p>
    </div>
</body>
```

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

.background-cover, .background-contain {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100vw;
    height: 50vh;
    color: white;
    text-shadow: 2px 2px 4px rgba(0,0,0,0.6);
    margin-bottom: 20px;
}

.background-cover {
    background: url('https://example.com/cover-image.jpg') no-repeat center center;
    background-size: cover;
}

.background-contain {
    background: url('https://example.com/contain-image.jpg') no-repeat center center;
    background-size: contain;
}
```

HTML:
Dos elementos div, cada uno con una clase que indica si el fondo debe usar cover o contain.

CSS:

- **Estilos Comunes**: Ambos div usan flexbox para centrar el texto dentro. Se configura el tamaño a 100vw (100% del ancho de la ventana) y 50vh (50% del alto de la ventana) para que ocupen una porción significativa de la pantalla, facilitando la demostración de cómo las imágenes de fondo se adaptan.

- **.background-cover:** Utiliza background-size: cover, lo que hace que la imagen de fondo se escale para cubrir completamente el área del div, posiblemente recortando partes de la imagen si las dimensiones de la imagen y el contenedor no coinciden.

- **.background-contain:** Utiliza background-size: contain, asegurando que la imagen completa esté visible sin recortes, ajustándose dentro del espacio disponible y dejando espacios vacíos si es necesario.

Uso de srcset y sizes en Imágenes:

srcset:

- Permite especificar un conjunto de imágenes para diferentes resoluciones, dejando que el navegador elija la más apropiada según las condiciones del dispositivo.

sizes:

- Define un conjunto de condiciones de media query y tamaños de imagen deseados, guiando al navegador sobre cómo seleccionar la imagen desde srcset.

```
<img src="imagen.jpg" srcset="imagen-320w.jpg 320w, imagen-480w.jpg 480w, imagen-800w.jpg 800w"
     sizes="(max-width: 320px) 280px, (max-width: 480px) 440px, 800px"
     alt="Imagen responsiva">
```

Ejercicio

- **Objetivo del Ejercicio:** Crear un layout responsivo que incluya un encabezado, un contenido principal y un pie de página que se adapte a móviles, tabletas y escritorios.

- **Requisitos:** El encabezado debe contener un logo y un menú de navegación que se convierta en un menú hamburguesa en móviles. El contenido principal debe tener dos columnas en escritorios y una sola columna en móviles.

```
<header>
    <img src="logo.png" alt="Logo" class="logo">
    <nav class="nav-main">
        <ul>
            <li><a href="#home">Inicio</a></li>
            <li><a href="#services">Servicios</a></li>
            <li><a href="#contact">Contacto</a></li>
        </ul>
    </nav>
</header>
```

```
<main>
    <section class="content">
        <article>
            <h1>Título Principal</h1>
            <p>Contenido principal...</p>
        </article>
        <aside>
            <h2>Noticias</h2>
            <p>Últimas noticias...</p>
        </aside>
    </section>
</main>
<footer>
    <p>Derechos reservados © 2024</p>
</footer>
```

```css
header img.logo {
    width: 100px;
    height: auto;
}

nav ul {
    list-style-type: none;
    padding: 0;
}

nav ul li {
    display: inline;
    margin-right: 10px;
}

@media (max-width: 600px) {
    nav ul li {
        display: block;
        margin-bottom: 5px;
    }
}

main .content {
    display: flex;
    flex-direction: row;
}
```

```css
@media (max-width: 800px) {
    main .content {
        flex-direction: column;
    }
}

footer {
    text-align: center;
    padding: 20px;
}
```

La Solución:

Encabezado: Se utiliza un logo que se escala de forma responsiva y un menú de navegación que cambia de un layout horizontal a un menú tipo hamburguesa en dispositivos móviles mediante Media Queries.

Contenido Principal: Se implementa un diseño de columna flexible que se adapta de dos columnas en escritorios a una sola columna en dispositivos más pequeños, usando flex-direction.

Pie de Página: Se centra el contenido y se asegura que el padding sea adecuado para todos los dispositivos.

Descripción del Ejercicio propuesto:

Crear una página simple que contenga un título, un párrafo descriptivo y una imagen. El diseño debe adaptarse para visualizarse de manera óptima tanto en móviles como en escritorios.

Requisitos específicos:

- El título debe aumentar su tamaño de fuente cuando se visualice en un escritorio y disminuir en un dispositivo móvil.
- El párrafo debe cambiar su alineación de texto de centrado en móviles a justificado en escritorios.
- La imagen debe ocupar el 100% del ancho de la pantalla en dispositivos móviles y el 50% en escritorios.