Tabla de contenidos

1. ¿Qué es CSS?

🎨 1. ¿Qué es CSS?

Definición

CSS (Cascading Style Sheets) o **Hojas de Estilo en Cascada** es un lenguaje de diseño utilizado para describir la presentación visual de documentos HTML. CSS controla el aspecto, formato y layout de las páginas web.

Propósito Principal

- **Separación de contenido y presentación**: HTML se encarga del contenido estructural, CSS de la apariencia visual
- **Reutilización**: Un archivo CSS puede aplicarse a múltiples páginas HTML
- **Mantenimiento**: Facilita la actualización del diseño sin modificar el HTML
- **Consistencia**: Garantiza un diseño uniforme en todo el sitio web

La Triada del Desarrollo Web Frontend

Evolución de CSS

Versión	Año	Características Principales
CSS 1	1996	Propiedades básicas de texto y color
CSS 2	1998	Posicionamiento, media types
CSS 2.1	2004	Correcciones y refinamientos
CSS 3	2005-presente	Módulos independientes, nuevas propiedades

---

🛠️ 2. Métodos de Incorporación de CSS

Existen tres formas principales de incorporar CSS en un documento HTML, cada una con sus ventajas y casos de uso específicos.

2.1 CSS Inline (En línea)

Se aplica directamente en el elemento HTML utilizando el atributo `style`.

Sintaxis

```
<elemento style="propiedad: valor; propiedad2: valor2;">Contenido</elemento>
```

Ejemplo Práctico

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>CSS Inline</title>
</head>
<body>
    <h1 style="color: blue; font-size: 36px; text-align: center;">
        Título con CSS Inline
    </h1>

    <p style="color: green; font-weight: bold; margin: 20px;">
        Este párrafo tiene estilos aplicados directamente.
    </p>

    <div style="background-color: yellow; padding: 15px; border: 2px solid red;">
        Contenedor con fondo amarillo y borde rojo.
    </div>
</body>
</html>
```

Ventajas del CSS Inline

- **Especificidad alta**: Tiene prioridad sobre otros estilos
- **Rapidez**: Útil para pruebas rápidas o cambios específicos
- **Sin archivos externos**: Todo está en el mismo documento

Desventajas del CSS Inline

- **No reutilizable**: Cada elemento debe estilizarse individualmente
- **Difícil mantenimiento**: Cambios requieren editar múltiples elementos
- **Código verbose**: Hace el HTML menos legible
- **Mala práctica**: Va contra la separación de responsabilidades

2.2 CSS Embebido (Interno)

Se define dentro del elemento `<style>` en la sección `<head>` del documento HTML.

Sintaxis

```
<head>
    <style>
        selector {
            propiedad: valor;
            propiedad2: valor2;
        }
    </style>
</head>
```

Ejemplo Práctico

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>CSS Embebido</title>
    <style>
        /* Estilo para todos los h1 */
        h1 {
            color: darkblue;
            font-size: 2.5em;
            text-align: center;
            font-family: Arial, sans-serif;
        }

        /* Estilo para todos los párrafos */
        p {
            color: #333333;
            line-height: 1.6;
            margin: 15px 0;
            font-size: 16px;
        }

        /* Estilo para elementos con clase especial */
        .destacado {
            background-color: #f0f8ff;
            padding: 20px;
            border-left: 4px solid #4CAF50;
            margin: 20px 0;
        }

        /* Estilo para elemento con ID específico */
        #contenido-principal {
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
        }
    </style>
</head>
<body>
    <div id="contenido-principal">
        <h1>Título Principal</h1>

        <p>Este es un párrafo normal con los estilos definidos en la sección head.</p>

        <div class="destacado">
            <p>Este contenido está dentro de un div con la clase "destacado".</p>
        </div>

        <p>Otro párrafo que mantiene el estilo consistente.</p>
    </div>
</body>
</html>
```

Ventajas del CSS Embebido

- **Organización**: Estilos centralizados en un solo lugar del documento
- **Reutilización**: Un estilo se aplica a múltiples elementos
- **Especificidad**: Mayor control que CSS externo
- **Sin dependencias**: No requiere archivos adicionales

Desventajas del CSS Embebido

- **Limitado a una página**: No se puede reutilizar en otros documentos
- **Tamaño del archivo**: Aumenta el tamaño del HTML
- **Mantenimiento**: Difícil de mantener en sitios grandes
- **Carga**: El CSS se descarga con cada página

2.3 CSS Externo

Se define en archivos separados con extensión `.css` y se vincula al HTML mediante el elemento `<link>`.

Estructura de Archivos

```
proyecto/
│
├── index.html
├── sobre-nosotros.html
├── contacto.html
│
└── css/
    ├── estilos.css
    ├── normalize.css
    └── responsive.css
```

Sintaxis de Vinculación

```
<head>
    <link rel="stylesheet" href="ruta/al/archivo.css">
</head>
```

Ejemplo Práctico

Archivo: css/estilos.css

```css
/* Reset básico */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Estilos del body */
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    line-height: 1.6;
    color: #333;
    background-color: #f8f9fa;
}

/* Header principal */
.header {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    padding: 2rem 0;
    text-align: center;
}

.header h1 {
    font-size: 3rem;
    margin-bottom: 0.5rem;
    text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
}

.header p {
    font-size: 1.2rem;
    opacity: 0.9;
}

/* Contenedor principal */
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
}

/* Navegación */
.nav {
    background-color: #343a40;
    padding: 1rem 0;
}

.nav ul {
    list-style: none;
    display: flex;
    justify-content: center;
    gap: 2rem;
}

.nav a {
    color: white;
    text-decoration: none;
    font-weight: 500;
    transition: color 0.3s ease;
}

.nav a:hover {
    color: #007bff;
}

/* Contenido principal */
.main-content {
    background: white;
    margin: 2rem auto;
    padding: 2rem;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

/* Tarjetas */
```

```
.card {
    background: white;
    border-radius: 8px;
    padding: 1.5rem;
    margin-bottom: 1.5rem;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.card:hover {
    transform: translateY(-5px);
    box-shadow: 0 4px 20px rgba(0,0,0,0.15);
}

.card h2 {
    color: #495057;
    margin-bottom: 1rem;
    border-bottom: 2px solid #007bff;
    padding-bottom: 0.5rem;
}

/* Footer */
.footer {
    background-color: #343a40;
    color: white;
    text-align: center;
    padding: 2rem 0;
    margin-top: 3rem;
}
```

Archivo: index.html

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Externo - Ejemplo</title>
    <link rel="stylesheet" href="css/estilos.css">
</head>
<body>
    <header class="header">
        <div class="container">
            <h1>Mi Sitio Web</h1>
            <p>Ejemplo de CSS externo bien organizado</p>
        </div>
    </header>

    <nav class="nav">
        <div class="container">
            <ul>
                <li><a href="#inicio">Inicio</a></li>
                <li><a href="#servicios">Servicios</a></li>
                <li><a href="#sobre-nosotros">Sobre Nosotros</a></li>
                <li><a href="#contacto">Contacto</a></li>
            </ul>
        </div>
    </nav>

    <main class="container">
        <section class="main-content">
            <div class="card">
                <h2>Bienvenido</h2>
                <p>Este sitio web utiliza CSS externo para mantener la separación entre contenido y presentación.</p>
            </div>

            <div class="card">
                <h2>Ventajas del CSS Externo</h2>
                <ul>
                    <li>Reutilización en múltiples páginas</li>
                    <li>Fácil mantenimiento</li>
                    <li>Mejor organización del código</li>
                    <li>Carga en caché para mejor rendimiento</li>
                </ul>
            </div>
        </section>
    </main>

    <footer class="footer">
        <div class="container">
            <p>&copy; 2025 Mi Sitio Web. Todos los derechos reservados.</p>
        </div>
    </footer>
</body>
</html>
```

Ventajas del CSS Externo

- **Reutilización máxima**: Un archivo CSS puede usarse en múltiples páginas
- **Mantenimiento centralizado**: Cambios en un solo archivo afectan todo el sitio
- **Organización**: Separación clara entre contenido y presentación
- **Rendimiento**: Los archivos CSS se almacenan en caché del navegador
- **Colaboración**: Diferentes desarrolladores pueden trabajar en HTML y CSS por separado

Desventajas del CSS Externo

- **Solicitudes HTTP adicionales**: Requiere descargar archivos separados
- **Dependencia**: Páginas no funcionan correctamente sin los archivos CSS
- **Complejidad inicial**: Requiere organización de archivos

📜 3. Sintaxis y Estructura de Reglas CSS

3.1 Anatomía de una Regla CSS

```
selector {
    propiedad: valor;
    propiedad2: valor2;
}
```

Componentes:

1. **Selector**: Indica qué elementos HTML serán afectados
2. **Declaración**: Conjunto de propiedades y valores
3. **Propiedad**: Aspecto del elemento que se va a modificar
4. **Valor**: Cómo se va a modificar la propiedad

3.2 Ejemplo Detallado

```
/* Selector de elemento */
h1 {
    color: #2c3e50;         /* Color del texto */
    font-size: 2.5rem;      /* Tamaño de fuente */
    text-align: center;     /* Alineación del texto */
    margin-bottom: 1rem;    /* Margen inferior */
}

/* Selector de clase */
.destacado {
    background-color: #f8f9fa; /* Color de fondo */
    padding: 20px;             /* Espaciado interno */
    border: 1px solid #dee2e6;  /* Borde */
    border-radius: 5px;        /* Bordes redondeados */
}

/* Selector de ID */
#contenido-principal {
    max-width: 1200px;         /* Ancho máximo */
    margin: 0 auto;            /* Centrado horizontal */
}
```

3.3 Reglas de Sintaxis

Comentarios

```
/* Comentario de una línea */

/* Comentario de
    múltiples líneas
*/
```

Sensibilidad a mayúsculas

```
/* Correcto */
color: red;
background-color: blue;

/* Incorrecto */
COLOR: red;          /* Las propiedades deben estar en minúsculas */
Background-Color: blue; /* Inconsistente */
```

Punto y coma

```css
/* Obligatorio después de cada declaración (excepto la última) */
h1 {
    color: blue;      /* Punto y coma requerido */
    font-size: 2rem;  /* Punto y coma requerido */
    text-align: center /* Opcional en la última declaración */
}
```

Espacios en blanco

```css
/* Formato recomendado para legibilidad */
.card {
    margin: 20px;
    padding: 15px;
    background-color: white;
}

/* También válido, pero menos legible */
.card{margin:20px;padding:15px;background-color:white;}
```

3.4 Valores y Unidades Comunes

Colores

```css
.ejemplos-colores {
    color: red;                    /* Nombre del color */
    background-color: #ff0000;     /* Hexadecimal */
    border-color: rgb(255, 0, 0);   /* RGB */
    outline-color: rgba(255, 0, 0, 0.5); /* RGBA con transparencia */
}
```

Unidades de Medida

```css
.ejemplos-unidades {
    /* Unidades absolutas */
    font-size: 16px;      /* Píxeles */
    margin: 1cm;          /* Centímetros */

    /* Unidades relativas */
    width: 50%;           /* Porcentaje del contenedor padre */
    font-size: 1.2em;     /* Relativo al tamaño de fuente del padre */
    padding: 2rem;        /* Relativo al tamaño de fuente del root (html) */
    height: 50vh;         /* 50% de la altura del viewport */
    width: 80vw;          /* 80% del ancho del viewport */
}
```

Palabras Clave

```css
.palabras-clave {
    display: block;       /* Valores específicos de la propiedad */
    position: relative;
    text-align: center;

    /* Valores globales */
    color: inherit;       /* Hereda del elemento padre */
    margin: initial;      /* Valor inicial de la propiedad */
    padding: unset;       /* Resetea a inherit o initial */
}
```

✨ 5. Buenas Prácticas

Organización de Código

```
/* 1. Comentarios descriptivos */
/* ===== RESET Y GENERALES ===== */

/* 2. Agrupación lógica */
/* ===== LAYOUT PRINCIPAL ===== */

/* 3. Orden alfabético de propiedades */
.elemento {
    background-color: white;
    border: 1px solid #ccc;
    color: #333;
    font-size: 16px;
    margin: 10px;
    padding: 15px;
}
```

Nombres de Clases Descriptivos

```
/* Buenas prácticas */
.nav-primary { }
.btn-submit { }
.card-product { }

/* Evitar */
.red-text { }
.big-font { }
.left { }
```

Evitar !important

```
/* Preferible: Usar especificidad correcta */
.container .nav .nav-item {
    color: blue;
}

/* Evitar: Uso innecesario de !important */
.nav-item {
    color: blue !important;
}
```

1.1. Introducción a las Unidades de Medida

📏 1. Introducción a las Unidades de Medida

Las **unidades de medida** en CSS determinan el tamaño de elementos, espaciado, tipografía y posicionamiento. Elegir la unidad correcta es crucial para crear diseños que sean **flexibles**, **accesibles** y **responsive**.

¿Por qué son importantes las unidades?

- **Flexibilidad**: Permiten diseños que se adaptan a diferentes dispositivos
- **Accesibilidad**: Respetan las preferencias del usuario (tamaño de fuente)
- **Mantenibilidad**: Facilitan cambios globales de escala
- **Rendimiento**: Optimizan el renderizado en diferentes resoluciones

Clasificación General

```
UNIDADES CSS
├── ABSOLUTAS
│   ├── px (píxeles)
│   ├── cm (centímetros)
│   ├── mm (milímetros)
│   ├── in (pulgadas)
│   ├── pt (puntos)
│   └── pc (picas)
└── RELATIVAS
    ├── % (porcentaje)
    ├── em (relativo al font-size del elemento)
    ├── rem (relativo al font-size del root)
    ├── vh (viewport height)
    ├── vw (viewport width)
    ├── vmin (viewport minimum)
    ├── vmax (viewport maximum)
    ├── ex (altura de la letra 'x')
    └── ch (ancho del carácter '0')
```

📏 2. Unidades Absolutas

Las unidades absolutas tienen un **tamaño físico fijo** que no cambia según el contexto.

2.1 Píxeles (px)

El píxel es la unidad absoluta más común en el desarrollo web.

Características

- **1px = 1/96 de pulgada** (en la mayoría de dispositivos)
- **Precisión**: Ideal para bordes, sombras y elementos que requieren control exacto
- **Limitaciones**: No se escala con las preferencias del usuario

Ejemplos Prácticos

```css
/* Casos ideales para píxeles */
.elemento {
    border: 1px solid #ccc;         /* Bordes finos */
    box-shadow: 0 2px 4px rgba(0,0,0,0.1); /* Sombras precisas */
    outline: 2px solid #007bff;     /* Contornos de accesibilidad */
}

/* Íconos y elementos pequeños */
.icono {
    width: 24px;
    height: 24px;
}

/* Media queries para breakpoints */
@media (max-width: 768px) {
    .sidebar {
        width: 300px;
    }
}
```

Ejemplo HTML Completo

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Unidades en Píxeles</title>
    <style>
        .container {
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
        }

        .card {
            border: 1px solid #dee2e6;
            border-radius: 8px;
            box-shadow: 0 2px 8px rgba(0,0,0,0.1);
            padding: 20px;
            margin-bottom: 20px;
        }

        .avatar {
            width: 48px;
            height: 48px;
            border-radius: 50%;
            border: 2px solid #007bff;
            background-color: #e3f2fd;
            display: inline-block;
            margin-right: 10px;
        }

        .divider {
            height: 1px;
            background-color: #eee;
            margin: 16px 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="card">
            <div class="avatar"></div>
            <strong>Usuario</strong>
            <div class="divider"></div>
            <p>Los píxeles son perfectos para elementos que necesitan dimensiones exactas como avatares, iconos y bordes.</p>
        </div>
    </div>
</body>
</html>
```

2.2 Otras Unidades Absolutas

Centímetros (cm), Milímetros (mm), Pulgadas (in)

```
/* Principalmente para medios impresos */
@media print {
    .pagina {
        width: 21cm;      /* A4 width */
        height: 29.7cm;   /* A4 height */
        margin: 2.54cm;   /* 1 inch margins */
    }

    .logo {
        width: 50mm;
        height: 20mm;
    }
}
```

Puntos (pt) y Picas (pc)

```
/* Tradicionalmente para tipografía impresa */
@media print {
    body {
        font-size: 12pt;      /* Tamaño estándar de impresión */
    }

    h1 {
        font-size: 24pt;
    }

    .columna {
        width: 6pc;           /* 1pc = 12pt */
    }
}
```

📏 3. Unidades Relativas

Las unidades relativas cambian según el **contexto** (elemento padre, tamaño de fuente, viewport, etc.).

3.1 Porcentaje (%)

Se calcula como un **porcentaje del elemento padre**.

Casos de Uso

```
/* Layouts responsivos */
.contenedor {
    width: 100%;        /* Ocupa todo el ancho del padre */
}

.sidebar {
    width: 25%;         /* 1/4 del contenedor padre */
}

.main-content {
    width: 75%;         /* 3/4 del contenedor padre */
}

/* Posicionamiento */
.centrado {
    position: absolute;
    top: 50%;           /* 50% desde arriba */
    left: 50%;          /* 50% desde la izquierda */
    transform: translate(-50%, -50%); /* Centrado perfecto */
}

/* Backgrounds */
.hero {
    background-size: cover;
    background-position: center 30%; /* 30% desde arriba */
}
```

Ejemplo Práctico Completo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Unidades en Porcentaje</title>
    <style>
        * {
            box-sizing: border-box;
        }

        .layout {
            width: 100%;
            min-height: 100vh;
            display: flex;
        }

        .sidebar {
            width: 20%;
            background-color: #343a40;
            color: white;
            padding: 20px;
        }

        .main {
            width: 80%;
            padding: 20px;
        }

        .progress-bar {
            width: 100%;
            height: 20px;
            background-color: #e9ecef;
            border-radius: 10px;
            overflow: hidden;
        }

        .progress-fill {
            height: 100%;
            background-color: #28a745;
            width: 65%;       /* 65% de progreso */
            transition: width 0.3s ease;
        }

        .grid {
            display: flex;
            gap: 2%;
        }

        .col {
            background-color: #f8f9fa;
            padding: 15px;
            border-radius: 4px;
        }

        .col-4 { width: 32%; }  /* (100% - 4%) / 3 */
        .col-6 { width: 49%; }  /* (100% - 2%) / 2 */
        .col-12 { width: 100%; }

        @media (max-width: 768px) {
            .layout {
                flex-direction: column;
            }

            .sidebar,
            .main {
                width: 100%;
            }

            .col-4,
            .col-6 {
                width: 100%;
                margin-bottom: 2%;
            }
        }
    </style>
```

```
</head>
<body>
    <div class="layout">
        <aside class="sidebar">
            <h3>Sidebar (20%)</h3>
            <p>Esta barra lateral ocupa el 20% del ancho total.</p>
        </aside>

        <main class="main">
            <h1>Contenido Principal (80%)</h1>

            <div style="margin: 20px 0;">
                <h3>Barra de Progreso</h3>
                <div class="progress-bar">
                    <div class="progress-fill"></div>
                </div>
                <p>Progreso: 65%</p>
            </div>

            <h3>Grid con Porcentajes</h3>
            <div class="grid">
                <div class="col col-4">
                    <h4>Columna 1 (32%)</h4>
                    <p>Primera columna de tres.</p>
                </div>
                <div class="col col-4">
                    <h4>Columna 2 (32%)</h4>
                    <p>Segunda columna de tres.</p>
                </div>
                <div class="col col-4">
                    <h4>Columna 3 (32%)</h4>
                    <p>Tercera columna de tres.</p>
                </div>
            </div>

            <div class="grid" style="margin-top: 20px;">
                <div class="col col-6">
                    <h4>Columna 1 (49%)</h4>
                    <p>Primera columna de dos.</p>
                </div>
                <div class="col col-6">
                    <h4>Columna 2 (49%)</h4>
                    <p>Segunda columna de dos.</p>
                </div>
            </div>
        </main>
    </div>
</body>
</html>
```

3.2 Em (em)

La unidad **em** es relativa al `font-size` del **elemento actual**.

Características

- **1em = tamaño de fuente del elemento**
- **Herencia**: Si el padre tiene `font-size: 20px` , entonces `1em = 20px` para el hijo
- **Compuesto**: Los valores em se multiplican en elementos anidados

Ejemplos Prácticos

```css
/* Espaciado proporcional al texto */
.articulo {
    font-size: 16px;      /* Base */
}

.articulo h1 {
    font-size: 2em;       /* 32px (16px × 2) */
    margin-bottom: 0.5em; /* 16px (32px × 0.5) */
}

.articulo p {
    font-size: 1em;       /* 16px */
    margin-bottom: 1em;   /* 16px */
    line-height: 1.5em;   /* 24px */
}

.articulo .small {
    font-size: 0.875em;   /* 14px (16px × 0.875) */
    padding: 0.5em 1em;   /* 7px 14px */
}
```

Problema del Compuesto

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Problema del Em Compuesto</title>
    <style>
        .lista {
            font-size: 16px;
        }

        .lista li {
            font-size: 0.9em;   /* 14.4px en primer nivel */
        }

        /* PROBLEMA: Se compone en elementos anidados */
        /* Nivel 1: 16px × 0.9 = 14.4px */
        /* Nivel 2: 14.4px × 0.9 = 12.96px */
        /* Nivel 3: 12.96px × 0.9 = 11.66px */
    </style>
</head>
<body>
    <ul class="lista">
        <li>Elemento nivel 1 (14.4px)
            <ul>
                <li>Elemento nivel 2 (12.96px)
                    <ul>
                        <li>Elemento nivel 3 (11.66px)</li>
                    </ul>
                </li>
            </ul>
        </li>
    </ul>
</body>
</html>
```

3.3 Rem (rem)

La unidad **rem** (root em) es relativa al `font-size` del elemento **root** (html).

Ventajas sobre Em

- **No se compone**: Siempre relativo al elemento raíz
- **Consistencia**: Predecible en todos los niveles de anidación
- **Escalabilidad**: Cambiar el font-size del html escala todo el diseño

Ejemplos Prácticos

```
/* Establecer base en el root */
html {
    font-size: 16px;      /* Base: 1rem = 16px */
}

/* Sistema de tipografía con rem */
h1 { font-size: 2.5rem; }     /* 40px */
h2 { font-size: 2rem; }       /* 32px */
h3 { font-size: 1.5rem; }     /* 24px */
h4 { font-size: 1.25rem; }    /* 20px */
p  { font-size: 1rem; }       /* 16px */
small { font-size: 0.875rem; } /* 14px */

/* Espaciado consistente */
.container {
    padding: 2rem;          /* 32px */
    margin-bottom: 3rem;    /* 48px */
}

.card {
    padding: 1.5rem;        /* 24px */
    margin-bottom: 2rem;    /* 32px */
    border-radius: 0.5rem;  /* 8px */
}

.btn {
    padding: 0.75rem 1.5rem; /* 12px 24px */
    font-size: 1rem;        /* 16px */
}
```

Sistema de Diseño Escalable

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Sistema de Diseño con REM</title>
    <style>
        /* Base escalable */
        html {
            font-size: 16px;    /* Default: 1rem = 16px */
        }

        /* Modo compacto */
        html.compact {
            font-size: 14px;    /* Compact: 1rem = 14px */
        }

        /* Modo amplio */
        html.large {
            font-size: 18px;    /* Large: 1rem = 18px */
        }

        /* Componentes que se escalan automáticamente */
        .header {
            padding: 2rem;
            font-size: 1.5rem;
        }

        .card {
            padding: 1rem;
            margin: 1rem;
            border-radius: 0.5rem;
            border: 1px solid #ddd;
        }

        .btn {
            padding: 0.5rem 1rem;
            font-size: 0.875rem;
            border-radius: 0.25rem;
            border: none;
            background: #007bff;
            color: white;
            cursor: pointer;
        }

        .text-small {
            font-size: 0.75rem;
            color: #666;
        }

        /* Grid system con rem */
        .container {
            max-width: 75rem;     /* 1200px en tamaño default */
            margin: 0 auto;
            padding: 0 1rem;
        }

        .row {
            display: flex;
            margin: 0 -0.5rem;
        }

        .col {
            flex: 1;
            padding: 0 0.5rem;
        }
    </style>
</head>
<body>
    <div class="container">
        <header class="header">
            <h1>Título Principal</h1>
        </header>

        <div class="row">
            <div class="col">
```

```
            <div class="card">
                <h3>Card 1</h3>
                <p>Contenido de ejemplo que se escala proporcionalmente.</p>
                <button class="btn">Acción</button>
                <p class="text-small">Texto pequeño adicional</p>
            </div>
        </div>

        <div class="col">
            <div class="card">
                <h3>Card 2</h3>
                <p>Todo se escala cambiando solo el font-size del html.</p>
                <button class="btn">Acción</button>
                <p class="text-small">Nota: Prueba cambiar las clases del html</p>
            </div>
        </div>
    </div>
</div>

<script>
    // Función para cambiar el tamaño base
    function changeSize(size) {
        document.documentElement.className = size;
    }
</script>
</body>
</html>
```

📺 4. Unidades Viewport

Las unidades viewport son relativas al **tamaño de la ventana del navegador**. Son fundamentales para crear diseños **truly responsive**.

4.1 Viewport Width (vw)

- **1vw = 1% del ancho del viewport**
- **100vw** = ancho completo de la ventana
- **Uso común**: Elementos de ancho completo, tipografía fluida

4.2 Viewport Height (vh)

- **1vh = 1% de la altura del viewport**
- **100vh** = altura completa de la ventana
- **Uso común**: Secciones de altura completa, modales

4.3 Viewport Minimum (vmin) y Maximum (vmax)

- **vmin**: 1% de la dimensión más pequeña del viewport
- **vmax**: 1% de la dimensión más grande del viewport
- **Uso**: Elementos que se adaptan a cualquier orientación

Ejemplos Prácticos Completos

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Unidades Viewport</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        /* Hero section - altura completa */
        .hero {
            height: 100vh;            /* Altura completa del viewport */
            width: 100vw;             /* Ancho completo del viewport */
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            display: flex;
            align-items: center;
            justify-content: center;
            color: white;
            text-align: center;
        }

        .hero h1 {
            font-size: 8vw;           /* Tipografía que se escala con el viewport */
            max-font-size: 80px;      /* Límite superior */
            min-font-size: 32px;      /* Límite inferior */
        }

        .hero p {
            font-size: 2vw;
            max-font-size: 24px;
            min-font-size: 16px;
            margin-top: 2vh;
        }

        /* Sidebar que se adapta */
        .layout {
            display: flex;
            min-height: 100vh;
        }

        .sidebar {
            width: 20vw;              /* 20% del ancho del viewport */
            min-width: 200px;         /* Ancho mínimo */
            background: #f8f9fa;
            padding: 2vh 1vw;
        }

        .main-content {
            flex: 1;
            padding: 2vh 2vw;
        }

        /* Modal centrado */
        .modal {
            position: fixed;
            top: 0;
            left: 0;
            width: 100vw;
            height: 100vh;
            background: rgba(0,0,0,0.8);
            display: flex;
            align-items: center;
            justify-content: center;
        }

        .modal-content {
            background: white;
            width: 90vw;              /* 90% del viewport */
            max-width: 600px;         /* Máximo 600px */
            height: 80vh;             /* 80% de la altura */
            max-height: 500px;        /* Máximo 500px */
            padding: 5vh 5vw;
```

```
    border-radius: 8px;
    overflow-y: auto;
}

/* Cards responsivos con viewport */
.cards-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(30vw, 1fr));
    gap: 2vw;
    padding: 2vw;
}

.card {
    background: white;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    padding: 3vh 2vw;
    min-height: 40vh;         /* Altura mínima relativa al viewport */
}

/* Elementos circulares adaptativos */
.circle {
    width: 20vmin;            /* Siempre circular independiente de orientación */
    height: 20vmin;
    border-radius: 50%;
    background: #007bff;
    margin: 0 auto;
}

/* Navegación sticky */
.nav {
    position: sticky;
    top: 0;
    height: 10vh;             /* 10% de la altura del viewport */
    background: white;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    display: flex;
    align-items: center;
    padding: 0 5vw;
}

/* Footer */
.footer {
    height: 20vh;
    background: #2c3e50;
    color: white;
    display: flex;
    align-items: center;
    justify-content: center;
}

/* Media queries para ajustes finos */
@media (max-width: 768px) {
    .hero h1 {
        font-size: 12vw;      /* Más grande en móviles */
    }

    .sidebar {
        width: 100vw;         /* Ancho completo en móviles */
        height: auto;
    }

    .layout {
        flex-direction: column;
    }

    .cards-grid {
        grid-template-columns: 1fr; /* Una columna en móviles */
        gap: 4vw;
    }

    .modal-content {
        width: 95vw;
        height: 90vh;
        padding: 2vh 2vw;
    }
```

```
        }
    </style>
</head>
<body>
    <!-- Hero Section -->
    <section class="hero">
        <div>
            <h1>Título Principal</h1>
            <p>Tipografía que se escala con el viewport</p>
        </div>
    </section>

    <!-- Navegación -->
    <nav class="nav">
        <h2>Navegación Sticky (10vh)</h2>
    </nav>

    <!-- Layout principal -->
    <div class="layout">
        <aside class="sidebar">
            <h3>Sidebar (20vw)</h3>
            <p>Esta sidebar se adapta al ancho del viewport.</p>
        </aside>

        <main class="main-content">
            <h2>Contenido Principal</h2>

            <!-- Elemento circular adaptativo -->
            <div class="circle" style="margin: 4vh 0;"></div>
            <p style="text-align: center;">Círculo de 20vmin (siempre circular)</p>

            <!-- Grid de cards -->
            <div class="cards-grid">
                <div class="card">
                    <h3>Card 1</h3>
                    <p>Las cards se adaptan automáticamente usando unidades viewport.</p>
                </div>
                <div class="card">
                    <h3>Card 2</h3>
                    <p>Altura mínima de 40vh y ancho adaptativo con grid.</p>
                </div>
                <div class="card">
                    <h3>Card 3</h3>
                    <p>Sistema completamente responsive sin media queries complejas.</p>
                </div>
            </div>
        </main>
    </div>

    <!-- Footer -->
    <footer class="footer">
        <p>Footer con altura de 20vh</p>
    </footer>

    <!-- Modal (oculto por defecto) -->
    <div class="modal" style="display: none;">
        <div class="modal-content">
            <h2>Modal Responsivo</h2>
            <p>Este modal se adapta automáticamente a cualquier tamaño de pantalla usando unidades viewport.</p>
            <p>Ancho: 90vw (máx 600px)</p>
            <p>Alto: 80vh (máx 500px)</p>
        </div>
    </div>
</body>
</html>
```

➗ 5. Funciones Modernas de CSS

CSS moderno incluye funciones que permiten **cálculos dinámicos** y **valores adaptativos** más sofisticados.

5.1 Función calc()

Permite realizar **cálculos matemáticos** combinando diferentes unidades.

Sintaxis y Reglas

```
/* Sintaxis básica */
calc(expresión)

/* Reglas importantes */
1. Espacios obligatorios alrededor de + y -
    ✅ calc(100% - 20px)
    ❌ calc(100%-20px)

2. Se pueden combinar diferentes unidades
    ✅ calc(50vw + 2rem - 10px)

3. Anidación permitida
    ✅ calc(100% - calc(2em + 5px))

4. Soporta operaciones: +, -, *, /
```

Casos de Uso Prácticos

```
/* Layout con header y footer fijos */
.main-content {
    height: calc(100vh - 120px);  /* Vista completa menos header(60px) y footer(60px) */
    overflow-y: auto;
}

/* Sidebar con espacio para contenido */
.sidebar {
    width: calc(25% - 20px);      /* 25% menos margen */
}

/* Centrado perfecto con calc */
.centered {
    position: absolute;
    top: calc(50% - 25px);        /* Centro menos la mitad de la altura */
    left: calc(50% - 50px);       /* Centro menos la mitad del ancho */
    width: 100px;
    height: 50px;
}

/* Grid dinámico */
.grid-item {
    width: calc((100% - 40px) / 3);  /* 3 columnas con 20px de gap */
}

/* Tipografía fluida básica */
h1 {
    font-size: calc(24px + 2vw);      /* Mínimo 24px + escalado */
}

/* Padding proporcional */
.container {
    padding: calc(2rem + 5vw);        /* Responsive padding */
}
```

5.2 Función clamp()

Establece un valor **mínimo**, **preferido** y **máximo**. Es perfecta para tipografía fluida.

Sintaxis

```
clamp(mínimo, preferido, máximo)

/* Ejemplos */
font-size: clamp(16px, 4vw, 32px);
/* - Nunca menor a 16px
    - Escala con 4vw
    - Nunca mayor a 32px */

width: clamp(300px, 50%, 800px);
/* - Mínimo 300px
    - Preferido 50%
    - Máximo 800px */
```

Ejemplo Práctico Completo

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Tipografía Fluida con Clamp</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', sans-serif;
            line-height: 1.6;
        }

        /* Sistema de tipografía fluida */
        h1 {
            font-size: clamp(2rem, 6vw, 4rem);    /* 32px - escalable - 64px */
            line-height: 1.2;
            margin-bottom: clamp(1rem, 3vw, 2rem);
        }

        h2 {
            font-size: clamp(1.5rem, 4vw, 2.5rem);  /* 24px - escalable - 40px */
            margin-bottom: clamp(0.75rem, 2vw, 1.5rem);
        }

        h3 {
            font-size: clamp(1.25rem, 3vw, 2rem);    /* 20px - escalable - 32px */
            margin-bottom: clamp(0.5rem, 1.5vw, 1rem);
        }

        p {
            font-size: clamp(1rem, 2vw, 1.25rem);    /* 16px - escalable - 20px */
            margin-bottom: clamp(1rem, 2vw, 1.5rem);
            max-width: 70ch;                         /* Líneas legibles */
        }

        /* Container fluido */
        .container {
            width: clamp(320px, 90%, 1200px);      /* Responsive width */
            margin: 0 auto;
            padding: clamp(1rem, 5vw, 3rem);     /* Responsive padding */
        }

        /* Cards con dimensiones fluidas */
        .card-grid {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(clamp(250px, 30%, 400px), 1fr));
            gap: clamp(1rem, 3vw, 2rem);
            margin: clamp(2rem, 5vw, 4rem) 0;
        }

        .card {
            background: white;
            border-radius: clamp(8px, 1vw, 16px);
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
            padding: clamp(1rem, 3vw, 2rem);
            height: clamp(200px, 40vw, 300px);
        }

        /* Botones adaptativos */
        .btn {
            padding: clamp(0.5rem, 2vw, 1rem) clamp(1rem, 4vw, 2rem);
            font-size: clamp(0.875rem, 2vw, 1.125rem);
            border-radius: clamp(4px, 0.5vw, 8px);
            border: none;
            background: #007bff;
            color: white;
            cursor: pointer;
            transition: background-color 0.3s ease;
        }
```

```
        .btn:hover {
            background: #0056b3;
        }

        /* Espaciado de secciones */
        section {
            margin-bottom: clamp(3rem, 8vw, 6rem);
        }

        /* Hero section responsive */
        .hero {
            text-align: center;
            padding: clamp(4rem, 12vw, 8rem) 0;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
        }

        .hero h1 {
            font-size: clamp(2.5rem, 8vw, 5rem);    /* Título hero más grande */
        }

        /* Navegación adaptive */
        nav {
            padding: clamp(0.5rem, 2vw, 1rem) 0;
            background: white;
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }

        .nav-container {
            display: flex;
            justify-content: space-between;
            align-items: center;
            max-width: 1200px;
            margin: 0 auto;
            padding: 0 clamp(1rem, 3vw, 2rem);
        }

        .logo {
            font-size: clamp(1.25rem, 3vw, 1.75rem);
            font-weight: bold;
        }

        /* Media query para casos extremos */
        @media (max-width: 480px) {
            .card-grid {
                grid-template-columns: 1fr;
            }
        }
    </style>
</head>
<body>
    <nav>
        <div class="nav-container">
            <div class="logo">Logo Fluido</div>
            <button class="btn">Botón Adaptativo</button>
        </div>
    </nav>

    <section class="hero">
        <div class="container">
            <h1>Título Hero Fluido</h1>
            <p>Esta tipografía se escala perfectamente en cualquier dispositivo usando clamp().</p>
            <button class="btn" style="margin-top: 2rem;">Call to Action</button>
        </div>
    </section>

    <section class="container">
        <h2>Sistema de Tipografía Fluida</h2>
        <p>
            Este sistema utiliza clamp() para crear una tipografía que se adapta
            automáticamente al tamaño de la pantalla, manteniendo siempre la
            legibilidad y proporción adecuada.
        </p>

        <h3>Ventajas del Clamp</h3>
        <p>
```

```
            La función clamp() nos permite definir valores mínimos y máximos,
            evitando que el texto sea demasiado pequeño en móviles o demasiado
            grande en pantallas gigantes.
        </p>

        <div class="card-grid">
            <div class="card">
                <h3>Card Fluida 1</h3>
                <p>Las dimensiones de estas cards se adaptan automáticamente.</p>
                <button class="btn">Acción</button>
            </div>

            <div class="card">
                <h3>Card Fluida 2</h3>
                <p>Sin media queries complejas, solo usando clamp().</p>
                <button class="btn">Acción</button>
            </div>

            <div class="card">
                <h3>Card Fluida 3</h3>
                <p>Responsive design simplificado y mantenible.</p>
                <button class="btn">Acción</button>
            </div>
        </div>
    </section>
</body>
</html>
```

5.3 Funciones min() y max()

Función min()

Selecciona el **valor más pequeño** de una lista de valores.

```
/* Ancho que no exceda 600px o el 100% del contenedor */
.container {
    width: min(600px, 100%);
}

/* Equivalente a: */
.container {
    width: 100%;
    max-width: 600px;
}

/* Padding que se adapta pero no excede 2rem */
.element {
    padding: min(5vw, 2rem);
}
```

Función max()

Selecciona el **valor más grande** de una lista de valores.

```
/* Altura mínima de 500px o 50vh (el mayor) */
.hero {
    height: max(50vh, 500px);
}

/* Equivalente a: */
.hero {
    height: 50vh;
    min-height: 500px;
}

/* Font size que nunca sea menor a 16px */
p {
    font-size: max(16px, 2vw);
}
```

📊 6. Comparación y Casos de Uso

6.1 Tabla Comparativa de Unidades

Unidad	Relativa a	Mejor uso	Evitar en
px	Tamaño fijo	Bordes, sombras, íconos	Tipografía, layouts principales
%	Elemento padre	Layouts fluidos, posicionamiento	Tipografía base
em	Font-size del elemento	Componentes con espaciado proporcional	Elementos muy anidados
rem	Font-size del root	Sistema de tipografía, espaciado general	Componentes que necesitan escala local
vw/vh	Viewport	Hero sections, tipografía fluida	Elementos pequeños, textos largos
vmin/vmax	Dimensión min/max del viewport	Elementos cuadrados, orientación adaptable	Layouts específicos de orientación

6.2 Guía de Decisión

🎯 ¿Qué unidad usar?

Para Tipografía:

- **rem** → Sistema de tipografía base
- **clamp()** → Tipografía fluida responsive
- **em** → Componentes con escala local

Para Layouts:

- **%** → Columnas fluidas, grids
- **vh/vw** → Secciones de altura/ancho completo
- **min/max** → Contenedores con límites

Para Detalles:

- **px** → Bordes, sombras, elementos pequeños
- **em** → Espaciado interno de componentes

✨ 7. Buenas Prácticas

7.1 Sistema de Medidas Consistente

```
/* Variables CSS para consistencia */
:root {
    /* Escala de espaciado basada en rem */
    --space-xs: 0.25rem;    /* 4px */
    --space-sm: 0.5rem;     /* 8px */
    --space-md: 1rem;       /* 16px */
    --space-lg: 1.5rem;     /* 24px */
    --space-xl: 2rem;       /* 32px */
    --space-2xl: 3rem;      /* 48px */
    --space-3xl: 4rem;      /* 64px */

    /* Tipografía fluida */
    --font-size-xs: clamp(0.75rem, 1vw, 0.875rem);
    --font-size-sm: clamp(0.875rem, 1.5vw, 1rem);
    --font-size-base: clamp(1rem, 2vw, 1.125rem);
    --font-size-lg: clamp(1.125rem, 2.5vw, 1.25rem);
    --font-size-xl: clamp(1.25rem, 3vw, 1.5rem);
    --font-size-2xl: clamp(1.5rem, 4vw, 2rem);
    --font-size-3xl: clamp(2rem, 5vw, 3rem);

    /* Contenedores */
    --container-sm: min(540px, 100%);
    --container-md: min(720px, 100%);
    --container-lg: min(960px, 100%);
    --container-xl: min(1140px, 100%);
    --container-2xl: min(1320px, 100%);
}

/* Uso de las variables */
.card {
    padding: var(--space-lg);
    margin-bottom: var(--space-xl);
}

.container {
    width: var(--container-lg);
    margin: 0 auto;
    padding: 0 var(--space-md);
}

h1 { font-size: var(--font-size-3xl); }
h2 { font-size: var(--font-size-2xl); }
h3 { font-size: var(--font-size-xl); }
```

7.2 Combinaciones Efectivas

```
/* Combinando diferentes unidades estratégicamente */

/* Layout principal: % para estructura, rem para espacios */
.layout {
    display: grid;
    grid-template-columns: 25% 1fr;
    gap: 2rem;
    padding: 2rem;
    min-height: calc(100vh - 4rem);  /* Viewport menos padding */
}

/* Componente: rem para escala base, em para elementos internos */
.button {
    font-size: 1rem;          /* Base con rem */
    padding: 0.75em 1.5em;    /* Proporcional con em */
    border-radius: 0.5rem;    /* Consistente con rem */
    border: 2px solid;        /* Fino con px implícito */
}

/* Hero responsive: vw para impacto, clamp() para control */
.hero {
    height: 100vh;
    padding: clamp(2rem, 8vw, 6rem);
}

.hero h1 {
    font-size: clamp(2rem, 8vw, 5rem);
    margin-bottom: min(3rem, 6vw);
}

/* Cards adaptativas: combinación completa */
.card-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(min(300px, 100%), 1fr));
    gap: clamp(1rem, 4vw, 2rem);
}

.card {
    padding: clamp(1rem, 4vw, 2rem);
    border-radius: clamp(8px, 1vw, 16px);
    min-height: 40vh;
}
```

7.3 Testing y Debugging

Herramientas de Developer Tools

```css
/* CSS para debugging unidades */
[data-debug="units"] * {
    outline: 1px solid red;
    position: relative;
}

[data-debug="units"] *::before {
    content: attr(data-unit);
    position: absolute;
    top: -20px;
    left: 0;
    background: black;
    color: white;
    padding: 2px 4px;
    font-size: 10px;
    z-index: 1000;
}

/* Ejemplo de uso */
.test-element {
    width: clamp(200px, 50%, 400px);
}

/* En HTML: <div class="test-element" data-unit="clamp(200px, 50%, 400px)"> */
```

Checklist de Testing

✅ Lista de Verificación

- **Móvil (320px - 480px)**: ¿Es legible? ¿No hay overflow?
- **Tablet (481px - 768px)**: ¿Se adapta el layout?
- **Desktop (769px+)**: ¿Se aprovecha el espacio?
- **Accesibilidad**: ¿Respeta zoom del navegador?
- **Rendimiento**: ¿No hay recálculos excesivos?

💡 Puntos Clave

- **Combina unidades estratégicamente**: No hay una unidad "perfecta" para todo
- **Usa rem para sistemas escalables**: Especialmente tipografía y espaciado base
- **Aprovecha clamp() para responsive**: Reduce la necesidad de media queries
- **Reserva px para detalles**: Bordes, sombras y elementos que necesitan precisión
- **Testa en dispositivos reales**: Las unidades viewport pueden comportarse diferente
- **Mantén consistencia**: Usa variables CSS y sistemas predefinidos

2. Selectores CSS

🎯 1. Introducción a los Selectores CSS

Los **selectores CSS** son patrones que permiten identificar y aplicar estilos a elementos específicos del HTML. Son el puente entre el documento HTML y las reglas de estilo que queremos aplicar.

¿Por qué son importantes los selectores?

- **Precisión**: Permiten aplicar estilos exactamente donde los necesitamos
- **Eficiencia**: Evitan la repetición innecesaria de código
- **Mantenibilidad**: Facilitan la organización y actualización de estilos
- **Flexibilidad**: Ofrecen múltiples formas de seleccionar elementos

Sintaxis General

```
selector {
    propiedad: valor;
}
```

🎯 2. Selectores Básicos

2.1 Selector de Elemento (Type Selector)

Selecciona todos los elementos de un tipo específico en el documento.

Sintaxis

```
elemento {
    /* estilos */
}
```

Ejemplos Prácticos

```css
/* Aplicar estilos a todos los párrafos */
p {
    color: #333333;
    line-height: 1.6;
    margin-bottom: 1rem;
}

/* Estilos para todos los encabezados h1 */
h1 {
    font-size: 2.5rem;
    color: #2c3e50;
    font-weight: bold;
    margin-bottom: 2rem;
}

/* Estilos para todas las listas no ordenadas */
ul {
    list-style-type: disc;
    padding-left: 2rem;
    margin-bottom: 1rem;
}

/* Estilos para todos los enlaces */
a {
    color: #1619FF;
    text-decoration: none;
    transition: color 0.3s ease;
}

/* Estilos para todas las imágenes */
img {
    max-width: 100%;
    height: auto;
    display: block;
}
```

Ejemplo HTML de Aplicación

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Selectores de Elemento</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            line-height: 1.6;
            color: #333;
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
        }

        h1 {
            color: #1619FF;
            border-bottom: 3px solid #1619FF;
            padding-bottom: 10px;
        }

        p {
            margin-bottom: 1rem;
            text-align: justify;
        }

        a {
            color: #FA01FA;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1>Selectores de Elemento en Acción</h1>
    <p>Este párrafo recibe estilos del selector de elemento 'p'.</p>
    <p>Todos los párrafos del documento comparten estos estilos.</p>
    <p>Los <a href="#ejemplo">enlaces</a> también tienen sus propios estilos.</p>
</body>
</html>
```

2.2 Selector de Clase (Class Selector)

Selecciona elementos que tienen un atributo `class` específico. Es reutilizable y puede aplicarse a múltiples elementos.

Sintaxis

```
.nombre-clase {
    /* estilos */
}
```

Ejemplos Prácticos

```css
/* Clase para botones primarios */
.btn-primary {
    background-color: #1619FF;
    color: white;
    padding: 12px 24px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s ease;
}

.btn-primary:hover {
    background-color: #0d0f9a; /* Darker blue */
}

/* Clase para tarjetas de contenido */
.card {
    background-color: white;
    border: 1px solid #dee2e6;
    border-radius: 8px;
    padding: 20px;
    margin-bottom: 20px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

/* Clase para texto destacado */
.destacado {
    background-color: #e0f8a2;
    border-left: 4px solid #A9F00F;
    padding: 15px;
    color: #333;
}

/* Clase para centrar contenido */
.centrado {
    text-align: center;
}

/* Clase para ocultar elementos */
.oculto {
    display: none;
}

/* Múltiples clases pueden combinarse */
.texto-grande {
    font-size: 1.25rem;
}

.texto-bold {
    font-weight: bold;
}
```

Ejemplo HTML de Aplicación

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Selectores de Clase</title>
    <link rel="stylesheet" href="estilos.css">
</head>
<body>
    <div class="card">
        <h2 class="centrado">Título de la Tarjeta</h2>
        <p>Contenido normal de la tarjeta.</p>
        <p class="destacado">Este párrafo está destacado con una clase especial.</p>
        <p class="texto-grande texto-bold centrado">
            Este párrafo combina múltiples clases.
        </p>
        <button class="btn-primary">Botón de Acción</button>
    </div>

    <div class="card oculto">
        <p>Esta tarjeta está oculta.</p>
    </div>
</body>
</html>
```

2.3 Selector de ID

Selecciona un elemento único que tiene un atributo `id` específico. Debe ser único en todo el documento.

Sintaxis

```
#nombre-id {
    /* estilos */
}
```

Ejemplos Prácticos

```css
/* Encabezado principal único */
#header-principal {
    background: #1619FF;
    color: white;
    padding: 40px 0;
    text-align: center;
    position: sticky;
    top: 0;
    z-index: 100;
}

/* Navegación principal */
#nav-principal {
    background-color: #343a40;
    padding: 15px 0;
}

#nav-principal ul {
    list-style: none;
    display: flex;
    justify-content: center;
    gap: 30px;
    margin: 0;
    padding: 0;
}

/* Contenido principal de la página */
#contenido-principal {
    max-width: 1200px;
    margin: 40px auto;
    padding: 0 20px;
    min-height: calc(100vh - 200px);
}

/* Pie de página único */
#footer-principal {
    background-color: #2c3e50;
    color: white;
    text-align: center;
    padding: 30px 0;
    margin-top: 50px;
}

/* Formulario de contacto específico */
#formulario-contacto {
    background-color: #f8f9fa;
    padding: 30px;
    border-radius: 10px;
    max-width: 600px;
    margin: 0 auto;
}

/* Modal específico */
#modal-confirmacion {
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background: white;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 4px 20px rgba(0,0,0,0.3);
    z-index: 1000;
}
```

Ejemplo HTML de Aplicación

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Selectores de ID</title>
    <link rel="stylesheet" href="estilos.css">
</head>
<body>
    <header id="header-principal">
        <h1>Mi Sitio Web</h1>
        <p>Ejemplo de selectores ID</p>
    </header>

    <nav id="nav-principal">
        <ul>
            <li><a href="#inicio">Inicio</a></li>
            <li><a href="#servicios">Servicios</a></li>
            <li><a href="#contacto">Contacto</a></li>
        </ul>
    </nav>

    <main id="contenido-principal">
        <section id="seccion-inicio">
            <h2>Bienvenido</h2>
            <p>Este es el contenido principal de la página.</p>
        </section>

        <section id="seccion-contacto">
            <h2>Contacto</h2>
            <form id="formulario-contacto">
                <label for="nombre">Nombre:</label>
                <input type="text" id="nombre" name="nombre">
                <button type="submit">Enviar</button>
            </form>
        </section>
    </main>

    <footer id="footer-principal">
        <p>© 2025 Mi Sitio Web. Todos los derechos reservados.</p>
    </footer>
</body>
</html>
```

🎯 3. Selectores de Atributo

Los selectores de atributo permiten seleccionar elementos basándose en la presencia o valor de sus atributos.

3.1 Tipos de Selectores de Atributo

[atributo] - Presencia del atributo

```css
/* Selecciona todos los elementos que tienen el atributo title */
[title] {
    border-bottom: 1px dotted #999;
    cursor: help;
}

/* Selecciona todos los inputs que tienen el atributo required */
[required] {
    border-left: 3px solid #e74c3c;
}
```

[atributo="valor"] - Valor exacto

```
/* Selecciona inputs de tipo email */
[type="email"] {
    background-image: url('email-icon.png');
    background-repeat: no-repeat;
    background-position: right 10px center;
    padding-right: 35px;
}

/* Selecciona enlaces externos */
[target="_blank"] {
    color: #FA01FA;
}

[target="_blank"]::after {
    content: " 🔗";
    font-size: 0.8em;
}
```

[atributo^="valor"] - Comienza con

```
/* Selecciona enlaces que comienzan con https */
[href^="https"] {
    color: #A9F00F;
}

[href^="https"]::before {
    content: "🔒 ";
}

/* Selecciona imágenes que comienzan con "thumb" */
[src^="thumb"] {
    border: 2px solid #1619FF;
    border-radius: 4px;
}
```

[atributo$="valor"] - Termina con

```
/* Selecciona enlaces a PDFs */
[href$=".pdf"] {
    color: #FA01FA;
}

[href$=".pdf"]::after {
    content: " 📄";
}

/* Selecciona imágenes JPG */
[src$=".jpg"], [src$=".jpeg"] {
    filter: sepia(10%);
}
```

[atributo*="valor"] - Contiene

```
/* Selecciona elementos que contienen "error" en su clase */
[class*="error"] {
    color: #e74c3c;
    font-weight: bold;
}

/* Selecciona elementos que contienen "success" en su clase */
[class*="success"] {
    color: #A9F00F;
    font-weight: bold;
}
```

3.2 Ejemplo Práctico Completo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Selectores de Atributo</title>
    <style>
        /* Elementos con atributo title */
        [title] {
            border-bottom: 1px dotted #999;
            cursor: help;
        }

        /* Inputs requeridos */
        [required] {
            border-left: 3px solid #e74c3c;
        }

        /* Inputs válidos */
        [type="email"]:valid {
            border-color: #A9F00F;
        }

        /* Enlaces externos */
        [href^="http"]:not([href^="http://misitio.com"]) {
            color: #FA01FA;
        }

        [href^="http"]:not([href^="http://misitio.com"])::after {
            content: " ↗";
            font-size: 0.8em;
        }

        /* Archivos descargables */
        [href$=".pdf"]::after { content: "  PDF"; }
        [href$=".doc"]::after { content: "  DOC"; }
        [href$=".zip"]::after { content: "  ZIP"; }

        /* Clases que contienen estado */
        [class*="warning"] {
            background-color: #fff3cd;
            border: 1px solid #ffeaa7;
            padding: 10px;
            border-radius: 4px;
        }

        [class*="success"] {
            background-color: #e0f8a2;
            border: 1px solid #A9F00F;
            padding: 10px;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <h1>Ejemplos de Selectores de Atributo</h1>

    <p title="Este párrafo tiene un título">
        Pasa el mouse sobre este párrafo.
    </p>

    <form>
        <label for="email">Email (requerido):</label>
        <input type="email" id="email" required>

        <label for="nombre">Nombre:</label>
        <input type="text" id="nombre">
    </form>

    <div>
        <a href="http://google.com">Enlace externo a Google</a><br>
        <a href="documento.pdf">Descargar PDF</a><br>
        <a href="archivo.zip">Descargar ZIP</a><br>
        <a href="/pagina-interna.html">Enlace interno</a>
```

```
    </div>

    <div class="mensaje-warning">
        Este es un mensaje de advertencia.
    </div>

    <div class="mensaje-success">
        ¡Operación exitosa!
    </div>
</body>
</html>
```

🪄 4. Pseudo-selectores

Los pseudo-selectores permiten seleccionar elementos basándose en su estado o posición, no en su estructura HTML.

4.1 Pseudo-clases de Estado

:hover - Al pasar el mouse

```css
/* Efecto hover en botones */
.boton {
    background-color: #1619FF;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: all 0.3s ease;
}

.boton:hover {
    background-color: #0d0f9a;
    transform: translateY(-2px);
    box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}

/* Hover en enlaces */
a:hover {
    color: #FA01FA;
    text-decoration: underline;
}

/* Hover en tarjetas */
.card:hover {
    transform: scale(1.05);
    box-shadow: 0 8px 16px rgba(0,0,0,0.15);
}
```

:focus - Cuando el elemento tiene foco

```css
/* Estilos para inputs con foco */
input:focus {
    outline: none;
    border-color: #1619FF;
    box-shadow: 0 0 0 3px rgba(22, 25, 255, 0.25);
}

/* Foco en botones */
button:focus {
    outline: 2px solid #1619FF;
    outline-offset: 2px;
}

/* Foco visible para accesibilidad */
:focus-visible {
    outline: 2px solid #1619FF;
    outline-offset: 2px;
}
```

:active - Durante el clic

```css
.boton:active {
    transform: translateY(1px);
    box-shadow: 0 2px 4px rgba(0,0,0,0.2);
}

a:active {
    color: #d900d9; /* Darker Fuchsia */
}
```

:visited - Enlaces visitados

```css
a:visited {
    color: #6f42c1;
}

/* Mantener hover en enlaces visitados */
a:visited:hover {
    color: #FA01FA;
}
```

4.2 Pseudo-clases Estructurales

:first-child y :last-child

```css
/* Primer elemento de una lista */
li:first-child {
    font-weight: bold;
    color: #1619FF;
}

/* Último elemento de una lista */
li:last-child {
    border-bottom: none;
    margin-bottom: 0;
}

/* Primer párrafo de un artículo */
article p:first-child {
    font-size: 1.1em;
    font-weight: 500;
}
```

:nth-child()

```css
/* Filas alternas en tablas */
tr:nth-child(even) {
    background-color: #f8f9fa;
}

tr:nth-child(odd) {
    background-color: white;
}

/* Cada tercer elemento */
.item:nth-child(3n) {
    margin-right: 0;
}

/* Primeros 3 elementos */
.item:nth-child(-n+3) {
    border-top: 3px solid #1619FF;
}

/* Elementos después del 3ro */
.item:nth-child(n+4) {
    opacity: 0.7;
}
```

4.3 Pseudo-elementos

Los pseudo-elementos permiten estilizar partes específicas de un elemento.

::before y ::after

```css
/* Agregar contenido decorativo */
.quote::before {
    content: '"';
    font-size: 3em;
    color: #1619FF;
    line-height: 0;
    margin-right: 10px;
    vertical-align: -20px;
}

.quote::after {
    content: '"';
    font-size: 3em;
    color: #1619FF;
    line-height: 0;
    margin-left: 10px;
    vertical-align: -20px;
}

/* Iconos con pseudo-elementos */
.enlace-externo::after {
    content: " ↗";
    font-size: 0.8em;
    color: #6c757d;
}

.archivo-pdf::before {
    content: "📄 ";
}

/* Tooltip con pseudo-elementos */
.tooltip {
    position: relative;
    cursor: help;
}

.tooltip::after {
    content: attr(data-tooltip);
    position: absolute;
    bottom: 125%;
    left: 50%;
    transform: translateX(-50%);
    background-color: #333;
    color: white;
    padding: 8px 12px;
    border-radius: 4px;
    font-size: 14px;
    white-space: nowrap;
    opacity: 0;
    visibility: hidden;
    transition: opacity 0.3s, visibility 0.3s;
}

.tooltip:hover::after {
    opacity: 1;
    visibility: visible;
}
```

::first-letter y ::first-line

```css
/* Letra capital */
.articulo::first-letter {
    float: left;
    font-size: 3em;
    line-height: 1;
    margin-right: 8px;
    margin-top: 2px;
    font-weight: bold;
    color: #1619FF;
}

/* Primera línea destacada */
.articulo::first-line {
    font-weight: bold;
    color: #2c3e50;
}
```

4.4 Ejemplo Práctico Completo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Pseudo-selectores</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            color: #333;
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
        }

        /* Lista con estilos alternos */
        .lista-productos li {
            padding: 15px;
            border-bottom: 1px solid #eee;
            transition: background-color 0.3s ease;
        }

        .lista-productos li:nth-child(even) {
            background-color: #f8f9fa;
        }

        .lista-productos li:hover {
            background-color: #e3f2fd;
            cursor: pointer;
        }

        .lista-productos li:first-child {
            border-top: 3px solid #1619FF;
            font-weight: bold;
        }

        .lista-productos li:last-child {
            border-bottom: 3px solid #1619FF;
        }

        /* Botones con estados */
        .btn {
            display: inline-block;
            padding: 12px 24px;
            background-color: #1619FF;
            color: white;
            text-decoration: none;
            border-radius: 4px;
            transition: all 0.3s ease;
            margin: 5px;
        }

        .btn:hover {
            background-color: #0d0f9a;
            transform: translateY(-2px);
        }

        .btn:active {
            transform: translateY(0);
        }

        /* Artículo con pseudo-elementos */
        .articulo {
            background: white;
            padding: 20px;
            margin: 20px 0;
            border-radius: 8px;
```

```
        box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    }

    .articulo::first-letter {
        float: left;
        font-size: 3em;
        line-height: 1;
        margin-right: 8px;
        margin-top: 2px;
        color: #1619FF;
        font-weight: bold;
    }

    /* Tooltips */
    .tooltip {
        position: relative;
        cursor: help;
        border-bottom: 1px dotted #999;
    }

    .tooltip::after {
        content: attr(data-tooltip);
        position: absolute;
        bottom: 125%;
        left: 50%;
        transform: translateX(-50%);
        background-color: #333;
        color: white;
        padding: 8px 12px;
        border-radius: 4px;
        font-size: 14px;
        white-space: nowrap;
        opacity: 0;
        visibility: hidden;
        transition: opacity 0.3s, visibility 0.3s;
        z-index: 1000;
    }

    .tooltip:hover::after {
        opacity: 1;
        visibility: visible;
    }
</style>
</head>
<body>
    <h1>Ejemplos de Pseudo-selectores</h1>

    <section>
        <h2>Lista con Efectos Alternos</h2>
        <ul class="lista-productos">
            <li>Producto Premium - $99.99</li>
            <li>Producto Estándar - $59.99</li>
            <li>Producto Básico - $29.99</li>
            <li>Producto Económico - $19.99</li>
            <li>Producto Especial - $79.99</li>
        </ul>
    </section>

    <section>
        <h2>Botones con Estados</h2>
        <a href="#" class="btn">Hover sobre mí</a>
        <a href="#" class="btn">Otro botón</a>
        <a href="#" class="btn">Tercer botón</a>
    </section>

    <section>
        <h2>Artículo con Letra Capital</h2>
        <div class="articulo">
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris.</p>
        </div>
    </section>

    <section>
        <h2>Tooltips Interactivos</h2>
        <p>Este párrafo contiene <span class="tooltip" data-tooltip="Esta es información adicional">información con tooltip</span>
```

```
que se muestra al pasar el mouse.</p>
    </section>
</body>
</html>
```

⚖️ 5. Especificidad y Cascada

5.1 ¿Qué es la Especificidad?

La especificidad determina qué regla CSS se aplica cuando múltiples reglas coinciden con el mismo elemento. Se calcula mediante un sistema de puntos:

Sistema de Puntuación

```
Inline styles:      1000 puntos
IDs:                100 puntos
Clases/atributos:   10 puntos
Elementos:          1 punto
```

Ejemplos de Cálculo

```
/* Especificidad: 1 (1 elemento) */
p { color: black; }

/* Especificidad: 10 (1 clase) */
.destacado { color: blue; }

/* Especificidad: 100 (1 ID) */
#titulo { color: red; }

/* Especificidad: 11 (1 clase + 1 elemento) */
p.destacado { color: green; }

/* Especificidad: 101 (1 ID + 1 elemento) */
#contenido p { color: purple; }

/* Especificidad: 111 (1 ID + 1 clase + 1 elemento) */
#contenido p.destacado { color: orange; }
```

5.2 Reglas de la Cascada

1. **Importancia**: `!important` tiene la mayor prioridad
2. **Especificidad**: Mayor especificidad gana
3. **Orden**: En caso de empate, la última regla gana

Ejemplo Práctico

```
<style>
    /* Especificidad: 1 */
    p { color: black; }

    /* Especificidad: 10 */
    .texto { color: blue; }

    /* Especificidad: 100 */
    #parrafo { color: red; }

    /* Especificidad: 11 */
    p.texto { color: green; }

    /* Mayor especificidad: 111 */
    #parrafo.texto { color: purple; }
</style>

<p id="parrafo" class="texto">
    Este texto será PURPLE (especificidad más alta)
</p>
```

5.3 Mejores Prácticas de Especificidad

```
/* ❌ Evitar: Especificidad innecesariamente alta */
#header #nav ul li a.link-primary {
    color: blue;
}

/* ✅ Mejor: Usar clases específicas */
.nav-link-primary {
    color: blue;
}

/* ❌ Evitar: Abuso de !important */
.texto {
    color: red !important;
}

/* ✅ Mejor: Usar especificidad correcta */
.contenido .texto {
    color: red;
}
```

🧬 6. Herencia en CSS

6.1 Propiedades que se Heredan

Algunas propiedades CSS se heredan automáticamente de elementos padre a hijo:

```
/* Propiedades que SE HEREDAN */
body {
    font-family: Arial, sans-serif;  /* Se hereda */
    color: #333;                    /* Se hereda */
    line-height: 1.6;               /* Se hereda */
    font-size: 16px;                /* Se hereda */
}

/* Propiedades que NO SE HEREDAN */
.contenedor {
    width: 800px;                   /* NO se hereda */
    height: 600px;                  /* NO se hereda */
    border: 1px solid #ccc;         /* NO se hereda */
    margin: 20px;                   /* NO se hereda */
    padding: 15px;                  /* NO se hereda */
}
```

6.2 Controlando la Herencia

```
/* Valores especiales para controlar herencia */
.elemento {
    color: inherit;    /* Hereda del padre */
    margin: initial;   /* Valor inicial de la propiedad */
    padding: unset;    /* Se resetea a inherit o initial */
}

/* Ejemplo práctico */
.contenedor {
    color: blue;
    font-size: 18px;
}

.hijo-normal {
    /* Hereda color: blue y font-size: 18px */
}

.hijo-modificado {
    color: red;        /* Sobrescribe el color heredado */
    /* Mantiene font-size: 18px heredado */
}

.hijo-reset {
    color: initial;    /* Resetea a valor inicial (black) */
    font-size: unset;  /* Resetea a valor heredado (18px) */
}
```

✨ 7. Buenas Prácticas

Nomenclatura de Selectores

```
/* ✅ Descriptivos y semánticos */
.nav-primary { }
.btn-submit { }
.card-product { }
.form-validation-error { }

/* ❌ Basados en apariencia */
.red-text { }
.big-button { }
.left-sidebar { }
```

Organización de CSS

```
/* 1. Selectores de elemento */
html, body, h1, h2, p { }

/* 2. Clases de layout */
.container { }
.grid { }
.sidebar { }

/* 3. Componentes */
.button { }
.card { }
.modal { }

/* 4. Utilidades */
.hidden { }
.centered { }
.clearfix { }

/* 5. Estados */
.is-active { }
.is-loading { }
.has-error { }
```

Evitar Selectores Problemáticos

```
/* ❌ Evitar: Selectores demasiado específicos */
#header nav ul li a.active { }

/* ✅ Mejor: Clases específicas */
.nav-link-active { }

/* ❌ Evitar: Selectores universales innecesarios */
* { margin: 0; padding: 0; }

/* ✅ Mejor: Reset selectivo */
html, body, h1, h2, h3, p, ul, ol { margin: 0; padding: 0; }
```

3. Modelo de Cajas

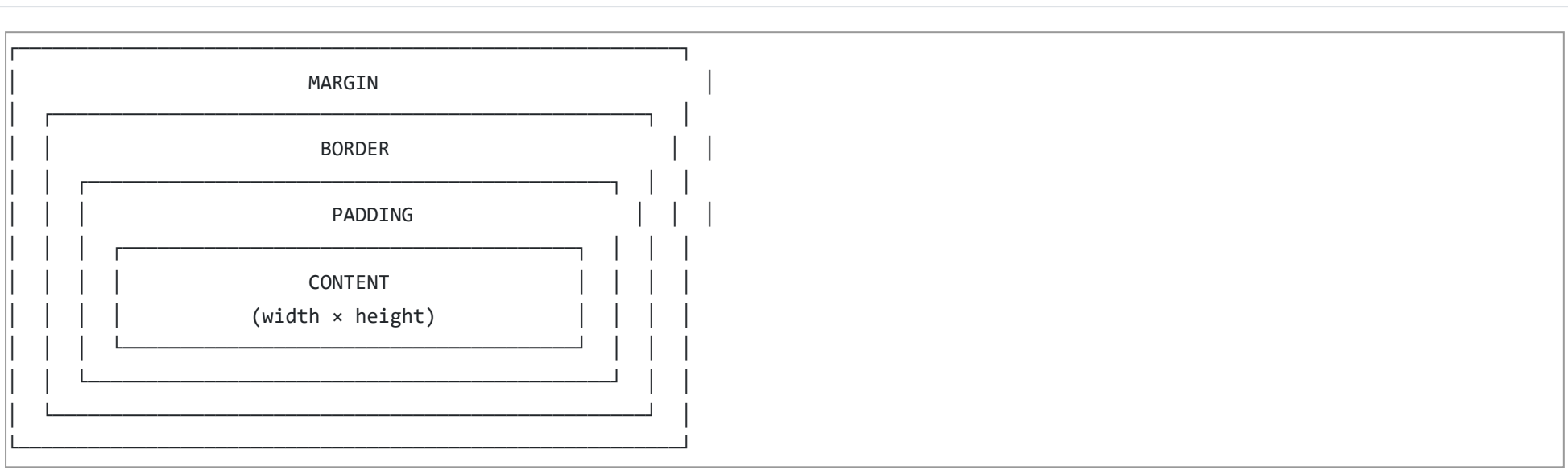
📦 1. Introducción al Modelo de Cajas

El **Modelo de Cajas (Box Model)** es uno de los conceptos fundamentales de CSS. Define cómo se calculan las dimensiones y el espaciado de todos los elementos HTML, tratando cada elemento como una caja rectangular compuesta por diferentes capas.

¿Por qué es importante el Box Model?

- **Control de dimensiones**: Determina el tamaño real de los elementos
- **Espaciado**: Controla la separación entre elementos
- **Layout**: Base para la construcción de diseños complejos
- **Debugging**: Esencial para resolver problemas de posicionamiento

Visualización del Modelo



🧱 2. Componentes del Modelo de Cajas

2.1 Content (Contenido)

El área donde se muestra el contenido real del elemento (texto, imágenes, etc.).

Propiedades Principales

```
.elemento {
    width: 300px;      /* Ancho del contenido */
    height: 200px;     /* Alto del contenido */
    min-width: 200px;  /* Ancho mínimo */
    max-width: 500px;  /* Ancho máximo */
    min-height: 100px; /* Alto mínimo */
    max-height: 400px; /* Alto máximo */
}
```

Ejemplo Práctico

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Content Area</title>
    <style>
        .contenedor {
            width: 300px;
            height: 200px;
            background-color: #e3f2fd;
            border: 2px solid #1619FF;
        }

        .contenedor-responsive {
            width: 80%;
            min-width: 250px;
            max-width: 600px;
            height: auto;
            min-height: 150px;
            background-color: #f3e5f5;
            border: 2px solid #FA01FA;
            margin: 20px 0;
        }

        .contenedor-flexible {
            width: 100%;
            height: 50vh;
            background-color: #e0f8a2;
            border: 2px solid #A9F00F;
        }
    </style>
</head>
<body>
    <div class="contenedor">
        <p>Esta caja tiene dimensiones fijas: 300px × 200px</p>
    </div>

    <div class="contenedor-responsive">
        <p>Esta caja es responsive: 80% de ancho con límites mínimos y máximos. El contenido determina la altura.</p>
    </div>

    <div class="contenedor-flexible">
        <p>Esta caja ocupa todo el ancho disponible y 50% de la altura del viewport.</p>
    </div>
</body>
</html>
```

2.2 Padding (Relleno)

Espacio transparente entre el contenido y el borde del elemento.

Sintaxis y Propiedades

```css
/* Padding individual */
.elemento {
    padding-top: 10px;
    padding-right: 15px;
    padding-bottom: 10px;
    padding-left: 15px;
}

/* Padding shorthand - 4 valores (top, right, bottom, left) */
.elemento {
    padding: 10px 15px 10px 15px;
}

/* Padding shorthand - 2 valores (vertical, horizontal) */
.elemento {
    padding: 10px 15px;
}

/* Padding uniforme */
.elemento {
    padding: 15px;
}
```

Ejemplos Prácticos

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Padding Examples</title>
    <style>
        .base-box {
            width: 200px;
            background-color: #fff3e0;
            border: 2px solid #ff9800;
            margin: 20px 0;
        }

        .sin-padding {
            padding: 0;
        }

        .padding-uniforme {
            padding: 20px;
        }

        .padding-asimetrico {
            padding: 10px 30px 15px 5px; /* top right bottom left */
        }

        .padding-vertical-horizontal {
            padding: 25px 10px; /* vertical horizontal */
        }

        /* Ejemplo de botón con padding */
        .boton {
            display: inline-block;
            padding: 12px 24px;
            background-color: #1619FF;
            color: white;
            text-decoration: none;
            border-radius: 4px;
            margin: 10px 5px;
        }

        .boton-pequeno {
            padding: 8px 16px;
            font-size: 14px;
        }

        .boton-grande {
            padding: 16px 32px;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <h2>Ejemplos de Padding</h2>

    <div class="base-box sin-padding">
        <p>Sin padding: el texto toca los bordes</p>
    </div>

    <div class="base-box padding-uniforme">
        <p>Padding uniforme de 20px en todos los lados</p>
    </div>

    <div class="base-box padding-asimetrico">
        <p>Padding asimétrico: top(10px) right(30px) bottom(15px) left(5px)</p>
    </div>

    <div class="base-box padding-vertical-horizontal">
        <p>Padding vertical(25px) horizontal(10px)</p>
    </div>

    <h3>Botones con Different Padding</h3>
    <a href="#" class="boton boton-pequeno">Botón Pequeño</a>
    <a href="#" class="boton">Botón Normal</a>
    <a href="#" class="boton boton-grande">Botón Grande</a>
```

```
</body>
</html>
```

2.3 Border (Borde)

Línea que rodea el padding y el contenido del elemento.

Propiedades del Border

```
/* Propiedades individuales */
.elemento {
    border-width: 2px;
    border-style: solid;
    border-color: #333;
}

/* Border shorthand */
.elemento {
    border: 2px solid #333;
}

/* Bordes individuales */
.elemento {
    border-top: 1px solid #ccc;
    border-right: 2px dashed #999;
    border-bottom: 3px dotted #666;
    border-left: 4px double #333;
}

/* Border radius para esquinas redondeadas */
.elemento {
    border-radius: 8px; /* Todas las esquinas */
    border-radius: 10px 5px; /* top-left/bottom-right, top-right/bottom-left */
    border-radius: 10px 5px 15px 20px; /* top-left, top-right, bottom-right, bottom-left */
}
```

Estilos de Border Disponibles

```
.border-styles {
    border-width: 3px;
    border-color: #333;
}

.solid { border-style: solid; }
.dashed { border-style: dashed; }
.dotted { border-style: dotted; }
.double { border-style: double; }
.groove { border-style: groove; }
.ridge { border-style: ridge; }
.inset { border-style: inset; }
.outset { border-style: outset; }
.none { border-style: none; }
.hidden { border-style: hidden; }
```

2.4 Margin (Margen)

Espacio transparente fuera del borde que separa el elemento de otros elementos.

Sintaxis y Propiedades

```css
/* Margin individual */
.elemento {
    margin-top: 20px;
    margin-right: 15px;
    margin-bottom: 20px;
    margin-left: 15px;
}

/* Margin shorthand - 4 valores (top, right, bottom, left) */
.elemento {
    margin: 20px 15px 20px 15px;
}

/* Margin shorthand - 2 valores (vertical, horizontal) */
.elemento {
    margin: 20px 15px;
}

/* Margin uniforme */
.elemento {
    margin: 20px;
}

/* Centrado horizontal */
.elemento {
    margin: 0 auto;
}

/* Margin negativo */
.elemento {
    margin-top: -10px; /* Superpone elementos */
}
```

📐 3. Box-sizing: Controlando el Cálculo de Dimensiones

3.1 ¿Qué es box-sizing?

La propiedad `box-sizing` determina cómo se calculan las dimensiones totales de un elemento.

Valores de box-sizing

content-box (valor por defecto)

```css
.elemento {
    box-sizing: content-box;
    width: 300px;
    padding: 20px;
    border: 5px solid #ccc;
}
/* Ancho total = 300px + 40px (padding) + 10px (border) = 350px */
```

border-box (más intuitivo)

```css
.elemento {
    box-sizing: border-box;
    width: 300px;
    padding: 20px;
    border: 5px solid #ccc;
}
/* Ancho total = 300px (incluye padding y border) */
```

3.2 Comparación Visual

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Box-sizing Comparison</title>
    <style>
        .container {
            display: flex;
            gap: 20px;
            margin: 20px;
            flex-wrap: wrap;
        }

        .box {
            width: 200px;
            height: 100px;
            padding: 20px;
            border: 5px solid #1619FF;
            background-color: #e3f2fd;
            margin: 10px 0;
        }

        .content-box {
            box-sizing: content-box;
        }

        .border-box {
            box-sizing: border-box;
        }

        .info {
            background-color: #f8f9fa;
            padding: 15px;
            border-radius: 4px;
            margin: 10px 0;
        }

        .demo-grid {
            display: grid;
            grid-template-columns: repeat(3, 1fr);
            gap: 20px;
            margin: 20px 0;
        }

        .grid-item {
            background-color: #fff3e0;
            border: 2px solid #ff9800;
            padding: 15px;
            box-sizing: border-box;
        }
    </style>
</head>
<body>
    <h2>Comparación de box-sizing</h2>

    <div class="container">
        <div>
            <h3>content-box (default)</h3>
            <div class="box content-box">
                Contenido
            </div>
            <div class="info">
                <strong>Ancho total:</strong> 200px + 40px (padding) + 10px (border) = <strong>250px</strong>
            </div>
        </div>

        <div>
            <h3>border-box</h3>
            <div class="box border-box">
                Contenido
            </div>
            <div class="info">
                <strong>Ancho total:</strong> <strong>200px</strong> (incluye padding y border)
```

```
            </div>
        </div>
    </div>
</body>
</html>
```

3.3 Reset Universal Recomendado

```
/* Reset box-sizing universal (muy recomendado) */
*,
*::before,
*::after {
    box-sizing: border-box;
}

/* Alternativa más específica */
html {
    box-sizing: border-box;
}

*,
*::before,
*::after {
    box-sizing: inherit;
}
```

🧱 4. Colapso de Márgenes

4.1 ¿Qué es el Colapso de Márgenes?

El colapso de márgenes ocurre cuando los márgenes verticales de elementos adyacentes se combinan en un solo margen.

Reglas del Colapso

- **Solo márgenes verticales** (top y bottom) colapsan
- **Solo en elementos en bloque** en flujo normal
- **El margen mayor prevalece** (no se suman)

4.2 Casos de Colapso de Márgenes

Caso 1: Elementos Hermanos Adyacentes

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Margin Collapse - Siblings</title>
    <style>
        .hermano1 {
            background-color: #e3f2fd;
            padding: 15px;
            margin-bottom: 30px; /* Margen inferior */
        }

        .hermano2 {
            background-color: #f3e5f5;
            padding: 15px;
            margin-top: 20px; /* Margen superior */
        }
        /* Resultado: separación de 30px (no 50px) */

        .sin-colapso .hermano2 {
            margin-top: 20px;
            border-top: 1px solid transparent; /* Previene colapso */
        }
    </style>
</head>
<body>
    <div class="hermano1">
        Elemento 1: margin-bottom: 30px
    </div>
    <div class="hermano2">
        Elemento 2: margin-top: 20px
    </div>
</body>
</html>
```

Caso 2: Elemento Padre e Hijo

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Margin Collapse - Parent Child</title>
    <style>
        .padre {
            background-color: #ffebee;
            margin-top: 20px;
            /* Sin padding-top ni border-top */
        }

        .hijo {
            background-color: #e8f5e8;
            margin-top: 30px;
            padding: 15px;
        }
        /* El margen del padre colapsa con el del hijo */

        .padre-sin-colapso {
            background-color: #fff3e0;
            margin-top: 20px;
            padding-top: 1px; /* Previene colapso */
        }
    </style>
</head>
<body>
    <div class="padre">
        <div class="hijo">
            Hijo con margin-top: 30px
        </div>
    </div>
</body>
</html>
```

4.3 Cómo Prevenir el Colapso de Márgenes

```css
/* Métodos para prevenir colapso de márgenes */

/* 1. Usando padding en lugar de margin */
.elemento {
    padding-top: 20px;
    padding-bottom: 20px;
}

/* 2. Usando border (aunque sea transparente) */
.elemento {
    border-top: 1px solid transparent;
}

/* 3. Usando overflow */
.elemento {
    overflow: hidden;
}

/* 4. Usando display flex o grid */
.contenedor {
    display: flex;
    flex-direction: column;
    gap: 20px; /* Espacio consistente sin colapso */
}

/* 5. Usando padding en el contenedor */
.contenedor {
    padding: 20px 0;
}

.contenedor > * {
    margin: 10px 0;
}
```

🐛 5. Herramientas de Debugging

5.1 Herramientas de Desarrollo del Navegador

```css
/* CSS temporal para visualizar el box model */
* {
    border: 1px solid red !important;
}

/* Más específico para debugging */
.debug * {
    border: 1px solid red;
    background-color: rgba(255, 0, 0, 0.1);
}

/* Outline no afecta el layout */
.debug * {
    outline: 1px solid blue;
}
```

5.2 Técnicas de Debugging

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Box Model Debugging</title>
    <style>
        /* Activar para debugging */
        .debug-mode * {
            outline: 1px solid red;
        }

        .debug-mode *:hover {
            background-color: rgba(255, 0, 0, 0.1);
        }

        /* Indicadores visuales de spacing */
        .debug-spacing {
            background-image:
                linear-gradient(to right, red 1px, transparent 1px),
                linear-gradient(to bottom, red 1px, transparent 1px);
            background-size: 20px 20px;
        }

        .card {
            width: 300px;
            padding: 20px;
            border: 2px solid #dee2e6;
            margin: 20px;
            background-color: white;
        }
    </style>
</head>
<body>
    <p>Abre las herramientas de desarrollo (F12) y inspecciona los elementos para ver el box model.</p>

    <div class="debug-mode">
        <div class="card">
            <h3>Tarjeta de Ejemplo</h3>
            <p>Esta tarjeta tiene padding, border y margin aplicados.</p>
        </div>
    </div>
</body>
</html>
```

✨ 6. Buenas Prácticas

6.1 Reset y Normalize CSS

```css
/* Box-sizing universal */
*,
*::before,
*::after {
    box-sizing: border-box;
}

/* Reset básico */
body {
    margin: 0;
    padding: 0;
}

/* Márgenes consistentes para elementos de texto */
h1, h2, h3, h4, h5, h6 {
    margin-top: 0;
    margin-bottom: 0.5em;
}

p {
    margin-top: 0;
    margin-bottom: 1em;
}

/* Último elemento sin margen inferior */
:last-child {
    margin-bottom: 0;
}
```

6.2 Sistemas de Espaciado Consistente

```css
/* Variables CSS para espaciado consistente */
:root {
    --spacing-xs: 4px;
    --spacing-sm: 8px;
    --spacing-md: 16px;
    --spacing-lg: 24px;
    --spacing-xl: 32px;
    --spacing-xxl: 48px;
}

/* Clases utilitarias */
.m-0 { margin: 0; }
.m-sm { margin: var(--spacing-sm); }
.m-md { margin: var(--spacing-md); }
.m-lg { margin: var(--spacing-lg); }

.p-0 { padding: 0; }
.p-sm { padding: var(--spacing-sm); }
.p-md { padding: var(--spacing-md); }
.p-lg { padding: var(--spacing-lg); }
```

6.3 Patrones Comunes

```css
/* Contenedor centrado */
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 var(--spacing-md);
}

/* Tarjeta básica */
.card {
    padding: var(--spacing-lg);
    border: 1px solid #dee2e6;
    border-radius: 8px;
    background-color: white;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

/* Botón con espaciado interno */
.btn {
    padding: var(--spacing-sm) var(--spacing-lg);
    border: 2px solid transparent;
    border-radius: 4px;
    display: inline-block;
    text-decoration: none;
    transition: all 0.3s ease;
}

/* Sección con espaciado vertical */
.section {
    padding: var(--spacing-xxl) 0;
}
```

4. Posicionamiento CSS

📍 1. Introducción al Posicionamiento CSS

El **posicionamiento CSS** determina cómo los elementos se colocan en el documento. Por defecto, los elementos siguen el "flujo normal" del documento, pero con CSS podemos cambiar este comportamiento para crear layouts complejos.

¿Por qué es importante el posicionamiento?

- **Control de layout**: Permite colocar elementos exactamente donde necesitamos
- **Efectos visuales**: Crea overlays, tooltips, menús desplegables
- **Responsive design**: Base para layouts adaptativos
- **Interactividad**: Elementos que aparecen/desaparecen dinámicamente

La Propiedad Position

```
.elemento {
    position: static | relative | absolute | fixed | sticky;
}
```

Propiedades de Posicionamiento

Una vez que un elemento tiene `position` diferente de `static`, podemos usar:

```
.elemento {
    top: valor;      /* Distancia desde arriba */
    right: valor;    /* Distancia desde la derecha */
    bottom: valor;   /* Distancia desde abajo */
    left: valor;     /* Distancia desde la izquierda */
    z-index: número; /* Orden de apilamiento */
}
```

📜 2. Position: Static (Por Defecto)

Características

- **Valor por defecto** de todos los elementos
- Sigue el **flujo normal** del documento
- **No se ve afectado** por top, right, bottom, left
- **No crea contexto de posicionamiento**

Ejemplo Práctico

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Position Static</title>
    <style>
        .container {
            width: 600px;
            margin: 20px auto;
            background-color: #f8f9fa;
            padding: 20px;
            border: 2px solid #dee2e6;
        }

        .box {
            width: 150px;
            height: 100px;
            margin: 10px;
            padding: 20px;
            background-color: #1619FF;
            color: white;
            text-align: center;
            line-height: 60px;
        }

        .static-box {
            position: static;
            /* Las siguientes propiedades NO tienen efecto */
            top: 50px;
            left: 100px;
            z-index: 999;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Position: Static (Default)</h2>
        <div class="box">Caja 1</div>
        <div class="box static-box">Caja 2 (Static)</div>
        <div class="box">Caja 3</div>
        <p>Las cajas siguen el flujo normal del documento. Las propiedades top, left no afectan elementos static.</p>
    </div>
</body>
</html>
```

3. Position: Relative

Características

- El elemento se posiciona **relativo a su posición original**
- **Mantiene su espacio** en el flujo del documento
- Puede usar top, right, bottom, left para moverse
- **Crea contexto de posicionamiento** para elementos hijos absolutos

Ejemplo Práctico

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Position Relative</title>
    <style>
        .container {
            width: 600px;
            margin: 20px auto;
            background-color: #f8f9fa;
            padding: 20px;
            border: 2px solid #dee2e6;
        }

        .box {
            width: 120px;
            height: 80px;
            margin: 10px;
            padding: 15px;
            background-color: #A9F00F;
            color: black;
            text-align: center;
            line-height: 50px;
            display: inline-block;
        }

        .relative-box {
            position: relative;
            top: 20px;
            left: 30px;
            background-color: #dc3545;
            color: white;
        }

        .relative-negative {
            position: relative;
            top: -15px;
            right: 20px;
            background-color: #fd7e14;
            color: white;
        }

        /* Ejemplo de contexto para elementos absolutos */
        .relative-parent {
            position: relative;
            width: 300px;
            height: 200px;
            background-color: #e3f2fd;
            border: 2px solid #1619FF;
            margin: 20px auto;
        }

        .absolute-child {
            position: absolute;
            top: 10px;
            right: 10px;
            width: 80px;
            height: 50px;
            background-color: #ff5722;
            color: white;
            text-align: center;
            line-height: 50px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Position: Relative</h2>

        <div class="box">Caja Normal</div>
        <div class="box relative-box">Relative (20px, 30px)</div>
        <div class="box">Caja Normal</div>
        <div class="box relative-negative">Relative (-15px, -20px)</div>
```

```
        <p>Las cajas relative se mueven desde su posición original pero mantienen su espacio en el flujo.</p>

        <h3>Relative como Contexto para Absolute</h3>
        <div class="relative-parent">
            <p style="margin: 0; padding: 10px;">Contenedor Relative</p>
            <div class="absolute-child">Absolute</div>
        </div>
    </div>
</body>
</html>
```

Casos de Uso Comunes

```
/* Pequeños ajustes de posición */
.icono {
    position: relative;
    top: 2px; /* Alinea verticalmente con el texto */
}

/* Crear contexto para elementos absolutos */
.card {
    position: relative; /* Permite que hijos absolutos se posicionen relativo a la card */
}

.card .badge {
    position: absolute;
    top: -10px;
    right: -10px;
}

/* Efectos hover */
.boton {
    position: relative;
    transition: all 0.3s ease;
}

.boton:hover {
    top: -2px; /* Efecto de elevación */
}
```

📦 4. Position: Absolute

Características

- Se posiciona **relativo al ancestro posicionado más cercano**
- Si no hay ancestro posicionado, se posiciona relativo al **viewport**
- **Se remueve del flujo** del documento (no ocupa espacio)
- Puede superponerse con otros elementos

Ejemplo Práctico Completo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Position Absolute</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: Arial, sans-serif;
            padding: 20px;
            background-color: #f5f5f5;
        }

        .demo-section {
            margin-bottom: 40px;
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        /* Ejemplo 1: Absolute sin contexto */
        .sin-contexto {
            width: 200px;
            height: 150px;
            background-color: #e3f2fd;
            border: 2px solid #1619FF;
            margin: 20px;
        }

        .absolute-viewport {
            position: absolute;
            top: 10px;
            right: 10px;
            width: 120px;
            height: 60px;
            background-color: #f44336;
            color: white;
            text-align: center;
            line-height: 60px;
            border-radius: 4px;
        }

        /* Ejemplo 2: Absolute con contexto */
        .con-contexto {
            position: relative; /* Crea contexto */
            width: 400px;
            height: 300px;
            background-color: #fff3e0;
            border: 2px solid #ff9800;
            margin: 20px auto;
        }

        .absolute-esquinas {
            position: absolute;
            width: 40px;
            height: 40px;
            background-color: #A9F00F;
            color: black;
            text-align: center;
            line-height: 40px;
            font-size: 12px;
            border-radius: 50%;
        }

        .top-left { top: 10px; left: 10px; }
        .top-right { top: 10px; right: 10px; }
        .bottom-left { bottom: 10px; left: 10px; }
```

```
.bottom-right { bottom: 10px; right: 10px; }
.center {
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: #FA01FA;
    color: white;
    width: 60px;
    height: 60px;
    line-height: 60px;
}

/* Ejemplo 3: Card con badge */
.card {
    position: relative;
    width: 300px;
    background: white;
    border-radius: 8px;
    padding: 20px;
    margin: 20px auto;
    box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    border: 1px solid #e0e0e0;
}

.badge {
    position: absolute;
    top: -10px;
    right: -10px;
    background-color: #ff5722;
    color: white;
    padding: 5px 10px;
    border-radius: 15px;
    font-size: 12px;
    font-weight: bold;
}

.close-btn {
    position: absolute;
    top: 10px;
    right: 10px;
    width: 25px;
    height: 25px;
    background-color: #f44336;
    color: white;
    border: none;
    border-radius: 50%;
    cursor: pointer;
    font-size: 14px;
    line-height: 1;
}

/* Ejemplo 4: Overlay */
.overlay-container {
    position: relative;
    width: 400px;
    height: 250px;
    background-image: url('data:image/svg+xml,<svg xmlns="http://www.w3.org/2000/svg" width="400" height="250" viewBox="0 0
400 250"><rect width="400" height="250" fill="%23e3f2fd"/><text x="200" y="125" text-anchor="middle" fill="%231619FF" font-
size="20">Imagen de Ejemplo</text></svg>');
    margin: 20px auto;
    border-radius: 8px;
    overflow: hidden;
}

.overlay {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: rgba(0, 0, 0, 0.7);
    color: white;
    display: flex;
    align-items: center;
    justify-content: center;
    opacity: 0;
```

```
        transition: opacity 0.3s ease;
    }

    .overlay-container:hover .overlay {
        opacity: 1;
    }

    /* Ejemplo 5: Tooltip */
    .tooltip-container {
        position: relative;
        display: inline-block;
        margin: 20px;
    }

    .tooltip-btn {
        padding: 10px 20px;
        background-color: #1619FF;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
    }

    .tooltip {
        position: absolute;
        bottom: 125%;
        left: 50%;
        transform: translateX(-50%);
        background-color: #333;
        color: white;
        padding: 8px 12px;
        border-radius: 4px;
        font-size: 14px;
        white-space: nowrap;
        opacity: 0;
        visibility: hidden;
        transition: all 0.3s ease;
    }

    .tooltip::after {
        content: '';
        position: absolute;
        top: 100%;
        left: 50%;
        transform: translateX(-50%);
        border: 5px solid transparent;
        border-top-color: #333;
    }

    .tooltip-container:hover .tooltip {
        opacity: 1;
        visibility: visible;
    }
    </style>
</head>
<body>
    <h1>Position: Absolute - Ejemplos Prácticos</h1>

    <div class="demo-section">
        <h2>1. Absolute sin Contexto (Relativo al Viewport)</h2>
        <div class="sin-contexto">
            <p style="padding: 20px;">Este div NO tiene position, por lo que el elemento absolute se posiciona relativo al viewport.
</p>
        </div>
        <div class="absolute-viewport">Viewport</div>
        <p><em>El elemento rojo se posiciona en la esquina superior derecha del viewport.</em></p>
    </div>

    <div class="demo-section">
        <h2>2. Absolute con Contexto (Relativo al Padre)</h2>
        <div class="con-contexto">
            <p style="padding: 20px; margin: 0;">Este contenedor tiene position: relative, creando contexto para elementos absolute.
</p>
            <div class="absolute-esquinas top-left">TL</div>
            <div class="absolute-esquinas top-right">TR</div>
            <div class="absolute-esquinas bottom-left">BL</div>
```

```
            <div class="absolute-esquinas bottom-right">BR</div>
            <div class="absolute-esquinas center">C</div>
        </div>
        <p><em>Los círculos se posicionan relativo al contenedor naranja, no al viewport.</em></p>
    </div>

    <div class="demo-section">
        <h2>3. Card con Badge y Botón de Cierre</h2>
        <div class="card">
            <div class="badge">NUEVO</div>
            <button class="close-btn">×</button>
            <h3>Título de la Tarjeta</h3>
            <p>Este es un ejemplo de cómo usar absolute para agregar elementos decorativos como badges y botones de cierre a
componentes.</p>
        </div>
    </div>

    <div class="demo-section">
        <h2>4. Overlay con Hover</h2>
        <div class="overlay-container">
            <div class="overlay">
                <div style="text-align: center;">
                </div>
            </div>
        </div>
        <p><em>El overlay cubre completamente la imagen usando top: 0, left: 0, right: 0, bottom: 0.</em></p>
    </div>

    <div class="demo-section">
        <h2>5. Tooltip Posicionado</h2>
        <div style="text-align: center;">
            <div class="tooltip-container">
                <button class="tooltip-btn">Hover para Tooltip</button>
                <div class="tooltip">Esta es información adicional</div>
            </div>
            <div class="tooltip-container">
                <button class="tooltip-btn">Otro Tooltip</button>
                <div class="tooltip">Tooltip con posicionamiento absolute</div>
            </div>
        </div>
        <p><em>Los tooltips se posicionan arriba del botón usando bottom: 125% y centrado con transform.</em></p>
    </div>
</body>
</html>
```

Técnicas Avanzadas de Centrado

```css
/* Centrado absoluto - Método 1: Transform */
.centrado-transform {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}

/* Centrado absoluto - Método 2: Márgenes (requiere dimensiones conocidas) */
.centrado-margins {
    position: absolute;
    top: 50%;
    left: 50%;
    width: 200px;
    height: 100px;
    margin-top: -50px;  /* -height/2 */
    margin-left: -100px; /* -width/2 */
}

/* Centrado absoluto - Método 3: Inset + Margin Auto */
.centrado-inset {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    width: 200px;
    height: 100px;
    margin: auto;
}
```

🔒 5. Position: Fixed

Características

- Se posiciona **relativo al viewport** (ventana del navegador)
- **No se mueve** al hacer scroll
- **Se remueve del flujo** del documento
- Ideal para headers fijos, botones flotantes, modales

Ejemplo Práctico

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Position Fixed</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
        }

        /* Header fijo */
        .header-fijo {
            position: fixed;
            top: 0;
            left: 0;
            right: 0;
            height: 60px;
            background-color: #2c3e50;
            color: white;
            display: flex;
            align-items: center;
            justify-content: space-between;
            padding: 0 20px;
            box-shadow: 0 2px 5px rgba(0,0,0,0.1);
            z-index: 1000;
        }

        .logo {
            font-size: 24px;
            font-weight: bold;
        }

        .nav {
            display: flex;
            gap: 20px;
        }

        .nav a {
            color: white;
            text-decoration: none;
            padding: 10px 15px;
            border-radius: 4px;
            transition: background-color 0.3s ease;
        }

        .nav a:hover {
            background-color: rgba(255,255,255,0.1);
        }

        /* Contenido principal con margen para el header fijo */
        .contenido {
            margin-top: 60px; /* Altura del header */
            padding: 20px;
        }

        .seccion {
            min-height: 100vh;
            padding: 40px 20px;
            border-bottom: 1px solid #eee;
        }

        .seccion:nth-child(even) {
            background-color: #f8f9fa;
        }

        /* Botón flotante */
```

```
.boton-flotante {
    position: fixed;
    bottom: 20px;
    right: 20px;
    width: 60px;
    height: 60px;
    background-color: #1619FF;
    color: white;
    border: none;
    border-radius: 50%;
    font-size: 24px;
    cursor: pointer;
    box-shadow: 0 4px 12px rgba(22, 25, 255, 0.3);
    transition: all 0.3s ease;
    z-index: 999;
}

.boton-flotante:hover {
    background-color: #0d0f9a;
    transform: scale(1.1);
}

/* Modal fijo */
.modal-overlay {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background-color: rgba(0, 0, 0, 0.5);
    display: flex;
    align-items: center;
    justify-content: center;
    z-index: 2000;
    opacity: 0;
    visibility: hidden;
    transition: all 0.3s ease;
}

.modal-overlay.active {
    opacity: 1;
    visibility: visible;
}

.modal {
    background: white;
    padding: 30px;
    border-radius: 8px;
    max-width: 500px;
    width: 90%;
    position: relative;
    transform: scale(0.7);
    transition: transform 0.3s ease;
}

.modal-overlay.active .modal {
    transform: scale(1);
}

.modal-close {
    position: absolute;
    top: 10px;
    right: 15px;
    background: none;
    border: none;
    font-size: 24px;
    cursor: pointer;
    color: #999;
}

/* Barra de progreso fija */
.progreso-fijo {
    position: fixed;
    top: 60px; /* Debajo del header */
    left: 0;
    height: 4px;
```

```
        background-color: #A9F00F;
        z-index: 999;
        transform-origin: left;
        transform: scaleX(0.3);
        transition: transform 0.3s ease;
    }

    /* Sidebar fijo */
    .sidebar-fijo {
        position: fixed;
        top: 60px;
        left: -300px;
        width: 300px;
        height: calc(100vh - 60px);
        background-color: white;
        border-right: 1px solid #ddd;
        padding: 20px;
        box-shadow: 2px 0 5px rgba(0,0,0,0.1);
        transition: left 0.3s ease;
        z-index: 998;
        overflow-y: auto;
    }

    .sidebar-fijo.active {
        left: 0;
    }

        .menu-toggle {
            background: none;
            border: none;
            color: white;
            font-size: 18px;
            cursor: pointer;
            padding: 10px;
        }
    </style>
</head>
<body>
    <!-- Header fijo -->
    <header class="header-fijo">
        <div style="display: flex; align-items: center; gap: 15px;">
            <button class="menu-toggle" onclick="toggleSidebar()">☰</button>
            <div class="logo">Mi Sitio</div>
        </div>
        <nav class="nav">
            <a href="#seccion1">Inicio</a>
            <a href="#seccion2">Servicios</a>
            <a href="#seccion3">Contacto</a>
        </nav>
    </header>

    <!-- Barra de progreso -->
    <div class="progreso-fijo" id="progreso"></div>

    <!-- Sidebar -->
    <aside class="sidebar-fijo" id="sidebar">
        <h3>Navegación</h3>
        <ul style="list-style: none; padding: 0;">
            <li style="margin-bottom: 10px;"><a href="#seccion1" style="text-decoration: none;">Sección 1</a></li>
            <li style="margin-bottom: 10px;"><a href="#seccion2" style="text-decoration: none;">Sección 2</a></li>
            <li style="margin-bottom: 10px;"><a href="#seccion3" style="text-decoration: none;">Sección 3</a></li>
            <li style="margin-bottom: 10px;"><a href="#seccion4" style="text-decoration: none;">Sección 4</a></li>
        </ul>
    </aside>

    <!-- Contenido principal -->
    <main class="contenido">
        <section class="seccion" id="seccion1">
            <h1>Sección 1 - Position Fixed</h1>
            <p>El header de arriba tiene <code>position: fixed</code>, por lo que permanece visible mientras haces scroll.</p>
            <p>El contenido principal tiene <code>margin-top: 60px</code> para no quedar oculto detrás del header fijo.</p>
            <p>Observa también el botón flotante en la esquina inferior derecha que permanece fijo en su posición.</p>
        </section>

        <section class="seccion" id="seccion2">
            <h2>Sección 2 - Elementos Fixed</h2>
```

```
            <p>Los elementos con position fixed son útiles para:</p>
            <ul>
                <li>Headers y navegación</li>
                <li>Notificaciones</li>
            </ul>
            <button onclick="abrirModal()" style="margin-top: 20px; padding: 10px 20px; background: #1619FF; color: white; border:
none; border-radius: 4px; cursor: pointer;">Abrir Modal</button>
        </section>

        <section class="seccion" id="seccion3">
            <h2>Sección 3 - Consideraciones</h2>
            <p>Al usar position fixed, ten en cuenta:</p>
            <ul>
                <li><strong>Z-index:</strong> Los elementos fixed necesitan z-index apropiado</li>
                <li><strong>Accesibilidad:</strong> Asegúrate de que no bloqueen contenido importante</li>
            </ul>
        </section>

        <section class="seccion" id="seccion4">
            <h2>Sección 4 - Final</h2>
            <p>Esta es la última sección. Observa cómo todos los elementos fixed mantienen su posición relativa al viewport sin
importar cuánto hagas scroll.</p>
            <p>El sidebar se puede abrir/cerrar con el botón de menú en el header.</p>
        </section>
    </main>

    </main>

    <!-- Botón flotante -->
    <button class="boton-flotante" onclick="scrollToTop()">↑</button>

    <!-- Modal -->
    <div class="modal-overlay" id="modalOverlay">
        <div class="modal">
            <button class="modal-close" onclick="cerrarModal()">×</button>
            <h3>Modal con Position Fixed</h3>
            <p>Este modal usa <code>position: fixed</code> para cubrir toda la pantalla y centrarse en el viewport.</p>
            <p>El overlay también es fixed y cubre todo el viewport para crear el efecto de modal.</p>
            <button onclick="cerrarModal()" style="margin-top: 20px; padding: 8px 16px; background: #6c757d; color: white; border:
none; border-radius: 4px; cursor: pointer;">Cerrar</button>
        </div>
    </div>

    <script>
        function toggleSidebar() {
            const sidebar = document.getElementById('sidebar');
            sidebar.classList.toggle('active');
        }

        function abrirModal() {
            document.getElementById('modalOverlay').classList.add('active');
        }

        function cerrarModal() {
            document.getElementById('modalOverlay').classList.remove('active');
        }

        function scrollToTop() {
            window.scrollTo({ top: 0, behavior: 'smooth' });
        }

        // Actualizar barra de progreso en scroll
        window.addEventListener('scroll', function() {
            const scrolled = (window.scrollY / (document.documentElement.scrollHeight - window.innerHeight)) * 100;
            document.getElementById('progreso').style.transform = `scaleX(${scrolled / 100})`;
        });
    </script>
</body>
</html>
```

📚 6. Z-index y Contextos de Apilamiento

¿Qué es Z-index?

El `z-index` controla el **orden de apilamiento** de elementos posicionados. Elementos con mayor z-index aparecen encima de elementos con menor z-index.

Reglas Importantes

```
/* Solo funciona en elementos posicionados */
.elemento {
    position: relative; /* absolute, fixed, sticky también funcionan */
    z-index: 10;
}

/* Los valores pueden ser positivos, negativos o auto */
.atras { z-index: -1; }
.normal { z-index: auto; /* valor por defecto */ }
.adelante { z-index: 999; }
```

Ejemplo Práctico de Z-index

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Z-index Examples</title>
    <style>
        .playground {
            position: relative;
            width: 500px;
            height: 400px;
            background-color: #f8f9fa;
            border: 2px solid #dee2e6;
            margin: 20px auto;
        }

        .caja {
            position: absolute;
            width: 120px;
            height: 120px;
            font-size: 18px;
            color: white;
            padding: 10px;
        }

        .caja1 {
            background-color: #e74c3c;
            top: 20px; left: 20px;
            z-index: 1;
        }

        .caja2 {
            background-color: #1619FF;
            top: 60px; left: 60px;
            z-index: 3;
        }

        .caja3 {
            background-color: #A9F00F;
            color: black;
            top: 100px; left: 100px;
            z-index: 2;
        }
    </style>
</head>
<body>
    <h2>Ejemplo de Z-index</h2>
    <div class="playground">
        <div class="caja caja1">Z: 1</div>
        <div class="caja caja2">Z: 3</div>
        <div class="caja caja3">Z: 2</div>
    </div>
    <p>La caja azul (z-index: 3) está encima, seguida de la verde (z-index: 2) y finalmente la roja (z-index: 1).</p>
</body>
</html>
```

4.1. Float y Display

🎨 1. Introducción a Float y Display

Antes de la llegada de **Flexbox** y **CSS Grid**, los desarrolladores dependían principalmente de **float** y **display** para crear layouts complejos. Aunque hoy en día son considerados métodos legacy para layout, siguen siendo importantes para casos específicos.

¿Por qué estudiar Float?

- **Contexto histórico**: Entender cómo se hacían layouts antes
- **Casos específicos**: Texto que rodea imágenes
- **Mantenimiento**: Muchos sitios existentes aún los usan
- **Fundamentos**: Base para entender conceptos modernos

¿Cuándo usar cada método?

```
MÉTODOS DE LAYOUT CSS
├── LEGACY (Pre-2010)
│   ├── Float + Clear
│   └── Display: inline-block
├── MODERNOS (2010+)
│   ├── Flexbox
│   └── CSS Grid
└── ESPECÍFICOS
    ├── Position (elementos superpuestos)
    └── Float (texto alrededor de imágenes)
```

➡️ 2. La Propiedad Float

¿Qué hace Float?

Float **"flota"** un elemento hacia la izquierda o derecha de su contenedor, permitiendo que el contenido fluya a su alrededor.

Valores de Float

```
.elemento {
    float: none;    /* Valor por defecto */
    float: left;    /* Flota a la izquierda */
    float: right;   /* Flota a la derecha */
    float: inherit; /* Hereda del padre */
}
```

Ejemplo Básico: Texto alrededor de Imagen

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Float Básico</title>
    <style>
        .articulo {
            max-width: 600px;
            margin: 20px auto;
            padding: 20px;
            background-color: #f8f9fa;
            border-radius: 8px;
            font-family: Arial, sans-serif;
            line-height: 1.6;
        }

        .imagen-izquierda {
            float: left;
            width: 200px;
            height: 150px;
            background-color: #1619FF;
            margin: 0 20px 20px 0;
            border-radius: 4px;
            display: flex;
            align-items: center;
            justify-content: center;
            color: white;
            font-weight: bold;
        }

        .imagen-derecha {
            float: right;
            width: 180px;
            height: 120px;
            background-color: #A9F00F;
            color: #000;
            margin: 0 0 20px 20px;
            border-radius: 4px;
            display: flex;
            align-items: center;
            justify-content: center;
            font-weight: bold;
        }

        .texto {
            text-align: justify;
        }

        .siguiente-seccion {
            background-color: #fff3cd;
            padding: 15px;
            margin-top: 20px;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <article class="articulo">
        <h1>Float: Texto Alrededor de Imágenes</h1>

        <div class="imagen-izquierda">Imagen Izquierda</div>

        <p class="texto">
            Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
        </p>

        <div class="imagen-derecha">Imagen Derecha</div>

        <p class="texto">
            Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo
```

```
inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.
        </p>

        <p class="texto">
            Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui
ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.
        </p>
    </article>

    <div class="siguiente-seccion">
        <h3>Sección Siguiente</h3>
        <p>Observa cómo esta sección puede verse afectada por los elementos flotados arriba si no se usa clear correctamente.</p>
    </div>
</body>
</html>
```

Comportamiento de Elementos Flotados

```
/* Cuando un elemento flota: */
.flotado {
    float: left;
    /* 1. Se remueve del flujo normal */
    /* 2. Se mueve hacia el lado especificado */
    /* 3. Otros elementos fluyen a su alrededor */
    /* 4. El contenedor padre puede "colapsar" */
}
```

🧹 3. La Propiedad Clear

¿Qué hace Clear?

Clear **previene** que un elemento se coloque al lado de elementos flotados, forzándolo a moverse debajo de ellos.

Valores de Clear

```
.elemento {
    clear: none;  /* Valor por defecto - permite floats a ambos lados */
    clear: left;  /* No permite floats a la izquierda */
    clear: right; /* No permite floats a la derecha */
    clear: both;  /* No permite floats en ningún lado */
}
```

Ejemplo de Clear en Acción

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Clear Examples</title>
    <style>
        .container {
            max-width: 800px;
            margin: 20px auto;
            padding: 20px;
            background-color: #f8f9fa;
            border-radius: 8px;
        }

        .caja {
            width: 150px;
            height: 100px;
            margin: 10px;
            padding: 20px;
            border-radius: 4px;
            color: white;
            text-align: center;
            line-height: 60px;
            font-weight: bold;
        }

        .float-left {
            float: left;
            background-color: #1619FF;
        }

        .float-right {
            float: right;
            background-color: #A9F00F;
            color: black;
        }

        .no-clear {
            background-color: #dc3545;
        }

        .clear-left {
            clear: left;
            background-color: #fd7e14;
        }

        .clear-right {
            clear: right;
            background-color: #6f42c1;
        }

        .clear-both {
            clear: both;
            background-color: #20c997;
        }

        .seccion {
            margin-bottom: 40px;
            padding: 20px;
            border: 2px solid #dee2e6;
            border-radius: 8px;
        }

        .clearfix::after {
            content: "";
            display: table;
            clear: both;
        }
    </style>
</head>
<body>
    <h1>Ejemplos de la Propiedad Clear</h1>
```

```
<div class="seccion">
    <h2>Sin Clear</h2>
    <div class="caja float-left">Float Left</div>
    <div class="caja float-right">Float Right</div>
    <div class="caja no-clear">Sin Clear</div>
    <p>La caja roja se coloca en el espacio disponible entre los elementos flotados.</p>
</div>

<div class="seccion clearfix">
    <h2>Clear: Left</h2>
    <div class="caja float-left">Float Left</div>
    <div class="caja float-right">Float Right</div>
    <div class="caja clear-left">Clear Left</div>
    <p>La caja naranja no permite floats a su izquierda, por lo que se mueve debajo del elemento flotado izquierdo.</p>
</div>

<div class="seccion clearfix">
    <h2>Clear: Right</h2>
    <div class="caja float-left">Float Left</div>
    <div class="caja float-right">Float Right</div>
    <div class="caja clear-right">Clear Right</div>
    <p>La caja morada no permite floats a su derecha, moviéndose debajo del elemento flotado derecho.</p>
</div>

<div class="seccion clearfix">
    <h2>Clear: Both</h2>
    <div class="caja float-left">Float Left</div>
    <div class="caja float-right">Float Right</div>
    <div class="caja clear-both">Clear Both</div>
    <p>La caja verde no permite floats en ningún lado, moviéndose completamente debajo de todos los elementos flotados.</p>
</div>
</body>
</html>
```

💥 4. Problema del Colapso de Contenedores

¿Qué es el Colapso?

Cuando todos los elementos hijos de un contenedor están flotados, el contenedor **"colapsa"** y pierde su altura.

Ejemplo del Problema

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Colapso de Contenedores</title>
    <style>
        .demo {
            margin: 20px auto;
            max-width: 600px;
        }

        .contenedor {
            background-color: #e3f2fd;
            border: 3px solid #1619FF;
            padding: 20px;
            margin-bottom: 20px;
        }

        .hijo-flotado {
            float: left;
            width: 150px;
            height: 100px;
            background-color: #ff5722;
            margin: 10px;
            border-radius: 4px;
            color: white;
            text-align: center;
            line-height: 100px;
        }

        .contenedor-sin-colapso {
            overflow: hidden; /* Una solución simple */
        }

        .clearfix::after {
            content: "";
            display: table;
            clear: both;
        }

        .siguiente-elemento {
            background-color: #fff3e0;
            padding: 15px;
            border: 2px solid #ff9800;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="demo">
        <h2>Problema: Contenedor Colapsado</h2>

        <div class="contenedor">
            <h3 style="margin-top: 0;">Contenedor con Colapso</h3>
            <div class="hijo-flotado">Float 1</div>
            <div class="hijo-flotado">Float 2</div>
            <div class="hijo-flotado">Float 3</div>
        </div>

        <div class="siguiente-elemento">
            <p>Este elemento puede superponerse con los flotados porque el contenedor colapsó.</p>
        </div>

        <h2>Solución 1: Overflow Hidden</h2>

        <div class="contenedor contenedor-sin-colapso">
            <h3 style="margin-top: 0;">Contenedor sin Colapso</h3>
            <div class="hijo-flotado">Float 1</div>
            <div class="hijo-flotado">Float 2</div>
            <div class="hijo-flotado">Float 3</div>
        </div>

        <div class="siguiente-elemento">
```

```
            <p>Ahora este elemento se posiciona correctamente debajo del contenedor.</p>
        </div>

        <h2>Solución 2: Clearfix</h2>

        <div class="contenedor clearfix">
            <h3 style="margin-top: 0;">Contenedor con Clearfix</h3>
            <div class="hijo-flotado">Float 1</div>
            <div class="hijo-flotado">Float 2</div>
            <div class="hijo-flotado">Float 3</div>
        </div>

        <div class="siguiente-elemento">
            <p>El clearfix también resuelve el problema del colapso.</p>
        </div>
    </div>
</body>
</html>
```

Soluciones al Colapso

1. Clearfix (Método Clásico)

```
.clearfix::after {
    content: "";
    display: table;
    clear: both;
}

/* Versión más robusta */
.clearfix::before,
.clearfix::after {
    content: "";
    display: table;
}

.clearfix::after {
    clear: both;
}
```

2. Overflow Hidden

```
.contenedor {
    overflow: hidden; /* Establece nuevo contexto de formato */
}
```

3. Display Flow-root (Moderno)

```
.contenedor {
    display: flow-root; /* Solución moderna y limpia */
}
```

🏛️ 5. Layout con Float (Método Legacy)

Grid System Básico con Float

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Layout con Float</title>
    <style>
        * {
            box-sizing: border-box;
        }

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

        .container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 0 20px;
        }

        .row {
            margin-left: -15px;
            margin-right: -15px;
        }

        .row::after {
            content: "";
            display: table;
            clear: both;
        }

        .col {
            float: left;
            padding: 0 15px;
            margin-bottom: 20px;
        }

        /* Sistema de columnas */
        .col-1 { width: 8.333%; }
        .col-2 { width: 16.666%; }
        .col-3 { width: 25%; }
        .col-4 { width: 33.333%; }
        .col-5 { width: 41.666%; }
        .col-6 { width: 50%; }
        .col-7 { width: 58.333%; }
        .col-8 { width: 66.666%; }
        .col-9 { width: 75%; }
        .col-10 { width: 83.333%; }
        .col-11 { width: 91.666%; }
        .col-12 { width: 100%; }

        /* Responsive */
        @media (max-width: 768px) {
            .col {
                width: 100% !important;
                float: none;
            }
        }

        /* Estilos visuales */
        .demo-box {
            background-color: #1619FF;
            color: white;
            padding: 20px;
            text-align: center;
            border-radius: 4px;
            min-height: 100px;
            display: flex;
            align-items: center;
```

```
        justify-content: center;
    }

    .demo-box:nth-child(2n) {
        background-color: #A9F00F;
        color: black;
    }

    .demo-box:nth-child(3n) {
        background-color: #FA01FA;
    }

    .header, .footer {
        background-color: #343a40;
        color: white;
        padding: 30px;
        text-align: center;
    }

    .sidebar {
        background-color: #6c757d;
    }

    .main-content {
        background-color: #17a2b8;
    }
    </style>
</head>
<body>
    <header class="header">
        <h1>Layout con Float - Sistema de Grid Legacy</h1>
    </header>

    <div class="container">
        <h2>Grid de 12 Columnas</h2>

        <div class="row">
            <div class="col col-12">
                <div class="demo-box">12 columnas (100%)</div>
            </div>
        </div>

        <div class="row">
            <div class="col col-6">
                <div class="demo-box">6 columnas (50%)</div>
            </div>
            <div class="col col-6">
                <div class="demo-box">6 columnas (50%)</div>
            </div>
        </div>

        <div class="row">
            <div class="col col-4">
                <div class="demo-box">4 columnas (33.33%)</div>
            </div>
            <div class="col col-4">
                <div class="demo-box">4 columnas (33.33%)</div>
            </div>
            <div class="col col-4">
                <div class="demo-box">4 columnas (33.33%)</div>
            </div>
        </div>

        <div class="row">
            <div class="col col-3">
                <div class="demo-box">3 columnas</div>
            </div>
            <div class="col col-3">
                <div class="demo-box">3 columnas</div>
            </div>
            <div class="col col-3">
                <div class="demo-box">3 columnas</div>
            </div>
            <div class="col col-3">
                <div class="demo-box">3 columnas</div>
            </div>
```

```
    </div>

    <h2>Layout Típico: Header, Sidebar, Content, Footer</h2>

    <div class="row">
        <div class="col col-3">
            <div class="demo-box sidebar">
                <div>
                    <h3>Sidebar</h3>
                    <p>Navegación secundaria</p>
                </div>
            </div>
        </div>
        <div class="col col-9">
            <div class="demo-box main-content">
                <div>
                    <h3>Contenido Principal</h3>
                    <p>Área principal del contenido</p>
                </div>
            </div>
        </div>
    </div>

    <div class="row">
        <div class="col col-8">
            <div class="demo-box">
                <div>
                    <h3>Artículo Principal</h3>
                    <p>Contenido del artículo</p>
                </div>
            </div>
        </div>
        <div class="col col-4">
            <div class="demo-box">
                <div>
                    <h3>Barra Lateral</h3>
                    <p>Contenido relacionado</p>
                </div>
            </div>
        </div>
    </div>

    <footer class="footer">
        <p>© 2025 Ejemplo de Layout con Float</p>
    </footer>
</body>
</html>
```

🖼️ 6. La Propiedad Display

Valores Principales de Display

```
.elemento {
    display: block;        /* Elemento de bloque */
    display: inline;       /* Elemento en línea */
    display: inline-block; /* Híbrido bloque/línea */
    display: none;         /* Oculto completamente */
    display: flex;         /* Contenedor flexible */
    display: grid;         /* Contenedor de grid */
    display: table;        /* Comportamiento de tabla */
    display: flow-root;    /* Nuevo contexto de formato */
}
```

Display: Block vs Inline vs Inline-block

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Display Examples</title>
    <style>
        .demo-container {
            max-width: 800px;
            margin: 20px auto;
            padding: 20px;
            background-color: #f8f9fa;
            border-radius: 8px;
        }

        .ejemplo {
            margin-bottom: 30px;
            padding: 20px;
            background-color: white;
            border-radius: 4px;
            border: 1px solid #dee2e6;
        }

        .elemento {
            background-color: #1619FF;
            color: white;
            padding: 10px;
            margin: 5px;
            border: 2px solid #0d0f9a;
        }

        .block {
            display: block;
        }

        .inline {
            display: inline;
        }

        .inline-block {
            display: inline-block;
        }

        .none {
            display: none;
        }

        /* Para mostrar diferencias */
        .width-height {
            width: 150px;
            height: 80px;
        }

        .text-container {
            background-color: #e9ecef;
            padding: 15px;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="demo-container">
        <h1>Propiedad Display</h1>

        <div class="ejemplo">
            <h2>Display: Block</h2>
            <div class="text-container">
                <span class="elemento block">Elemento 1 (block)</span>
                <span class="elemento block">Elemento 2 (block)</span>
                <span class="elemento block">Elemento 3 (block)</span>
            </div>
            <p><strong>Características:</strong> Ocupan todo el ancho disponible, comienzan en nueva línea, respetan width/height.
</p>
        </div>
```

```
        <div class="ejemplo">
            <h2>Display: Inline</h2>
            <div class="text-container">
                <span class="elemento inline">Elemento 1 (inline)</span>
                <span class="elemento inline">Elemento 2 (inline)</span>
                <span class="elemento inline">Elemento 3 (inline)</span>
            </div>
            <p><strong>Características:</strong> Se colocan en la misma línea, NO respetan width/height, solo margin/padding
horizontal.</p>
        </div>

        <div class="ejemplo">
            <h2>Display: Inline-block</h2>
            <div class="text-container">
                <span class="elemento inline-block width-height">Elemento 1</span>
                <span class="elemento inline-block width-height">Elemento 2</span>
                <span class="elemento inline-block width-height">Elemento 3</span>
            </div>
            <p><strong>Características:</strong> Se colocan en la misma línea PERO respetan width/height y todos los margin/padding.
</p>
        </div>

        <div class="ejemplo">
            <h2>Display: None</h2>
            <div class="text-container">
                <span class="elemento block">Elemento visible</span>
                <span class="elemento none">Elemento oculto</span>
                <span class="elemento block">Otro elemento visible</span>
            </div>
            <p><strong>Características:</strong> Elemento completamente removido del layout (no ocupa espacio).</p>
        </div>
    </div>
</body>
</html>
```

Display Table (Para Centrado Vertical Legacy)

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Display Table</title>
    <style>
        .table-container {
            display: table;
            width: 400px;
            height: 300px;
            background-color: #e3f2fd;
            border: 2px solid #1619FF;
            margin: 20px auto;
        }

        .table-cell {
            display: table-cell;
            vertical-align: middle;
            text-align: center;
            padding: 20px;
        }

        .content-box {
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        /* Layout de tabla completo */
        .table-layout {
            display: table;
            width: 100%;
            border-collapse: separate;
            border-spacing: 10px;
        }

        .table-row {
            display: table-row;
        }

        .table-cell-layout {
            display: table-cell;
            background-color: #f8f9fa;
            padding: 15px;
            border: 1px solid #dee2e6;
            border-radius: 4px;
        }

        .header-cell {
            background-color: #343a40;
            color: white;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h2>Display: Table para Centrado Vertical</h2>

    <div class="table-container">
        <div class="table-cell">
            <div class="content-box">
                <h3>Contenido Centrado</h3>
                <p>Este contenido está centrado vertical y horizontalmente usando display: table-cell.</p>
            </div>
        </div>
    </div>

    <h2>Layout de Tabla con Display</h2>

    <div class="table-layout">
        <div class="table-row">
            <div class="table-cell-layout header-cell">Nombre</div>
```

```
            <div class="table-cell-layout header-cell">Edad</div>
            <div class="table-cell-layout header-cell">Ciudad</div>
        </div>
        <div class="table-row">
            <div class="table-cell-layout">Juan Pérez</div>
            <div class="table-cell-layout">28</div>
            <div class="table-cell-layout">Madrid</div>
        </div>
        <div class="table-row">
            <div class="table-cell-layout">María García</div>
            <div class="table-cell-layout">32</div>
            <div class="table-cell-layout">Barcelona</div>
        </div>
    </div>
</body>
</html>
```

🤔 7. Cuándo Usar Cada Método

Tabla de Referencia

Método	Mejor Para	Evitar Para	Estado
Float	Texto alrededor de imágenes	Layouts principales	Legacy
Clear	Limpiar floats	Como método principal de layout	Específico
Inline-block	Navegación horizontal, botones	Layouts complejos	Semi-legacy
Display table	Centrado vertical legacy	Diseños responsivos	Legacy
Flexbox	Layouts 1D, alineación	Grids 2D complejos	Moderno
CSS Grid	Layouts 2D complejos	Alineación simple	Moderno

Cuándo Still Usar Float

```
/* ✅ USAR Float para: */

/* 1. Texto alrededor de imágenes (propósito original) */
.article-image {
    float: left;
    margin: 0 20px 20px 0;
}

/* 2. Elementos decorativos en texto */
.dropcap {
    float: left;
    font-size: 3em;
    line-height: 1;
    margin: 0 8px 0 0;
}

/* 3. Mantenimiento de código legacy */
.legacy-layout {
    float: left;
    width: 25%;
}
```

Migración a Métodos Modernos

```css
/* ❌ LEGACY: Float layout */
.old-container::after {
    content: "";
    display: table;
    clear: both;
}

.old-sidebar {
    float: left;
    width: 25%;
}

.old-main {
    float: right;
    width: 75%;
}

/* ✅ MODERNO: Flexbox */
.new-container {
    display: flex;
    gap: 20px;
}

.new-sidebar {
    flex: 0 0 25%;
}

.new-main {
    flex: 1;
}

/* ✅ MODERNO: CSS Grid */
.grid-container {
    display: grid;
    grid-template-columns: 1fr 3fr;
    gap: 20px;
}
```

🐛 8. Debugging de Float y Display

Herramientas de Debugging

```css
/* Debugging visual de floats */
.debug-floats * {
    outline: 1px solid red;
}

.debug-floats *:hover {
    background-color: rgba(255, 0, 0, 0.1);
}

/* Mostrar elementos con float */
[style*="float"] {
    border: 2px dashed blue !important;
}

/* Indicador visual para clearfix */
.clearfix::after {
    content: "CLEARFIX";
    background: yellow;
    color: black;
    font-size: 10px;
    padding: 2px 4px;
    position: absolute;
}
```

Problemas Comunes y Soluciones

```
/* Problema: Elementos inline-block con espacios */
.problema {
    display: inline-block; /* Hay espacios entre elementos */
}

/* Solución 1: font-size 0 en el padre */
.padre {
    font-size: 0;
}

.hijo {
    font-size: 16px;
    display: inline-block;
}

/* Solución 2: flexbox moderno */
.padre-moderno {
    display: flex;
    gap: 10px;
}
```

5. Tipografía Web

✍️ 1. Introducción a la Tipografía Web

¿Por qué es importante la tipografía?

La tipografía no es solo "hacer que el texto se vea bonito". Es fundamental para:

- **Legibilidad**: Facilitar la lectura del contenido
- **Jerarquía visual**: Guiar al usuario através del contenido
- **Experiencia de usuario**: Crear una experiencia agradable
- **Branding**: Reflejar la personalidad de la marca
- **Accesibilidad**: Asegurar que todos puedan leer el contenido

Anatomía de la Tipografía

```
ANATOMÍA TIPOGRÁFICA
├── FAMILIA (Font Family)
│   ├── Serif (con serifas)
│   ├── Sans-serif (sin serifas)
│   ├── Monospace (espaciado fijo)
│   └── Cursive/Fantasy (decorativas)
├── PESO (Font Weight)
│   ├── 100-300 (Light)
│   ├── 400 (Normal)
│   ├── 500-600 (Medium)
│   └── 700-900 (Bold)
├── ESTILO (Font Style)
│   ├── Normal
│   ├── Italic
│   └── Oblique
└── TAMAÑO (Font Size)
    ├── Absoluto (px, pt)
    └── Relativo (em, rem, %)
```

🔡 2. Propiedades de Fuente (Font Properties)

Font-Family: Familias de Fuentes

```
/* Pila de fuentes (font stack) */
.texto {
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
    /* Orden: preferida → alternativas → genérica */
}

/* Categorías genéricas */
.serif { font-family: serif; }         /* Times, Georgia */
.sans-serif { font-family: sans-serif; } /* Arial, Helvetica */
.monospace { font-family: monospace; }  /* Courier, Monaco */
.cursive { font-family: cursive; }     /* Script fonts */
.fantasy { font-family: fantasy; }     /* Decorative fonts */
```

Font-Size: Tamaño de Fuente

```
.tamaños {
    /* Unidades absolutas */
    font-size: 16px;   /* Pixeles */
    font-size: 12pt;   /* Puntos (print) */

    /* Unidades relativas */
    font-size: 1em;    /* Relativo al padre */
    font-size: 1.2rem; /* Relativo al root */
    font-size: 120%;   /* Porcentaje del padre */

    /* Valores predefinidos */
    font-size: small;
    font-size: medium;
    font-size: large;
    font-size: x-large;
}
```

Font-Weight: Peso de la Fuente

```
.pesos {
    font-weight: normal;   /* 400 */
    font-weight: bold;     /* 700 */
    font-weight: lighter;  /* Más ligero que el padre */
    font-weight: bolder;   /* Más pesado que el padre */

    /* Valores numéricos */
    font-weight: 100;      /* Thin */
    font-weight: 200;      /* Extra Light */
    font-weight: 300;      /* Light */
    font-weight: 400;      /* Normal */
    font-weight: 500;      /* Medium */
    font-weight: 600;      /* Semi Bold */
    font-weight: 700;      /* Bold */
    font-weight: 800;      /* Extra Bold */
    font-weight: 900;      /* Black */
}
```

Font-Style: Estilo de la Fuente

```
.estilos {
    font-style: normal;   /* Valor por defecto */
    font-style: italic;   /* Cursiva verdadera */
    font-style: oblique;  /* Inclinación artificial */
}
```

Ejemplo Completo: Sistema Tipográfico

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Sistema Tipográfico</title>
    <style>
        /* Reset y base */
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif;
            font-size: 16px;
            line-height: 1.6;
            color: #333;
            background-color: #f8f9fa;
            padding: 40px 20px;
        }

        .container {
            max-width: 800px;
            margin: 0 auto;
            background-color: white;
            padding: 40px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        /* Jerarquía tipográfica */
        h1 {
            font-size: 2.5rem;   /* 40px */
            font-weight: 700;
            line-height: 1.2;
            margin-bottom: 1rem;
            color: #1619FF;
        }

        h2 {
            font-size: 2rem;     /* 32px */
            font-weight: 600;
            line-height: 1.3;
            margin: 2rem 0 1rem 0;
            color: #FA01FA;
        }

        h3 {
            font-size: 1.5rem;   /* 24px */
            font-weight: 500;
            line-height: 1.4;
            margin: 1.5rem 0 0.75rem 0;
            color: #FA01FA;
        }

        h4 {
            font-size: 1.25rem;  /* 20px */
            font-weight: 500;
            line-height: 1.4;
            margin: 1rem 0 0.5rem 0;
            color: #FA01FA;
        }

        p {
            font-size: 1rem;     /* 16px */
            font-weight: 400;
            line-height: 1.6;
            margin-bottom: 1rem;
            text-align: justify;
        }

        /* Variaciones tipográficas */
```

```
.lead {
    font-size: 1.25rem;
    font-weight: 300;
    line-height: 1.5;
    color: #6c757d;
}

.small {
    font-size: 0.875rem; /* 14px */
    color: #6c757d;
}

.large {
    font-size: 1.125rem; /* 18px */
}

/* Énfasis y decoraciones */
.bold { font-weight: 700; }
.medium { font-weight: 500; }
.light { font-weight: 300; }
.italic { font-style: italic; }

/* Familias específicas */
.serif {
    font-family: Georgia, "Times New Roman", serif;
}

.mono {
    font-family: "Monaco", "Menlo", "Ubuntu Mono", monospace;
    background-color: #A9F00F;
    color: #000;
    padding: 2px 4px;
    border-radius: 3px;
    font-size: 0.9em;
}

/* Alineaciones */
.text-left { text-align: left; }
.text-center { text-align: center; }
.text-right { text-align: right; }
.text-justify { text-align: justify; }

/* Transformaciones */
.uppercase { text-transform: uppercase; }
.lowercase { text-transform: lowercase; }
.capitalize { text-transform: capitalize; }

/* Decoraciones */
.underline { text-decoration: underline; }
.line-through { text-decoration: line-through; }
.no-decoration { text-decoration: none; }

/* Espaciado de letras y palabras */
.letter-spacing { letter-spacing: 2px; }
.word-spacing { word-spacing: 4px; }

/* Ejemplo de código */
code {
    font-family: "Monaco", "Menlo", "Ubuntu Mono", monospace;
    background-color: #e9ecef;
    padding: 2px 6px;
    border-radius: 4px;
    font-size: 0.9em;
    color: #FA01FA;
}

pre {
    background-color: #f8f9fa;
    border: 1px solid #e9ecef;
    border-radius: 4px;
    padding: 15px;
    overflow-x: auto;
    margin: 1rem 0;
}

pre code {
```

```
            background: none;
            padding: 0;
            color: #333;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Sistema Tipográfico Completo</h1>

        <p class="lead">
            Un párrafo de introducción que destaca sobre el resto del contenido
            con un tamaño mayor y peso más ligero.
        </p>

        <h2>Jerarquía de Encabezados</h2>

        <h3>Encabezado de Tercer Nivel</h3>
        <p>
            Este es un párrafo normal que demuestra el <strong class="bold">texto en negrita</strong>,
            <em class="italic">texto en cursiva</em>, y texto con <span class="medium">peso medio</span>.
            También podemos usar <code>código inline</code> para destacar elementos técnicos.
        </p>

        <h4>Encabezado de Cuarto Nivel</h4>
        <p class="small">
            Este párrafo usa la clase <code>.small</code> para texto secundario o notas al pie.
        </p>

        <h3>Familias de Fuentes</h3>

        <p>Texto con fuente sans-serif (por defecto del sistema).</p>
        <p class="serif">Texto con fuente serif para una apariencia más tradicional.</p>
        <p class="mono">Texto monoespaciado ideal para código o datos tabulares.</p>

        <h3>Transformaciones de Texto</h3>

        <p class="uppercase">Este texto está en mayúsculas</p>
        <p class="lowercase">ESTE TEXTO ESTÁ EN MINÚSCULAS</p>
        <p class="capitalize">este texto tiene la primera letra en mayúscula</p>

        <h3>Decoraciones y Espaciado</h3>

        <p>Texto con <span class="underline">subrayado</span> y texto <span class="line-through">tachado</span>.</p>
        <p class="letter-spacing">TEXTO CON ESPACIADO DE LETRAS</p>
        <p class="word-spacing">Texto con espaciado de palabras aumentado</p>

        <h3>Alineaciones</h3>

        <p class="text-left">Texto alineado a la izquierda</p>
        <p class="text-center">Texto centrado</p>
        <p class="text-right">Texto alineado a la derecha</p>
        <p class="text-justify">
            Texto justificado que se distribuye uniformemente entre los márgenes
            izquierdo y derecho, creando líneas de igual longitud excepto posiblemente
            la última línea del párrafo.
        </p>

        <h3>Bloque de Código</h3>
        <pre><code>function saludar(nombre) {
    return `¡Hola, ${nombre}!`;
}

console.log(saludar('Mundo'));</code></pre>

        <p class="small">
            <em>Nota: Este sistema tipográfico está diseñado para ser escalable y mantener
            la legibilidad en diferentes dispositivos.</em>
        </p>
    </div>
</body>
</html>
```

3. Propiedades de Texto Avanzadas

Line-Height: Interlineado

```
.interlineado {
    line-height: normal;   /* Valor por defecto */
    line-height: 1.5;      /* Sin unidad - multiplicador */
    line-height: 24px;     /* Valor absoluto */
    line-height: 150%;     /* Porcentaje */
}

/* Mejores prácticas */
body {
    line-height: 1.6;      /* Buena legibilidad general */
}

h1, h2, h3 {
    line-height: 1.2;      /* Más apretado para títulos */
}
```

Text-Align: Alineación de Texto

```
.alineaciones {
    text-align: left;      /* Por defecto */
    text-align: right;     /* Derecha */
    text-align: center;    /* Centrado */
    text-align: justify;   /* Justificado */
    text-align: start;     /* Inicio (respeta dirección de texto) */
    text-align: end;       /* Final (respeta dirección de texto) */
}
```

Text-Transform: Transformación de Texto

```
.transformaciones {
    text-transform: none;       /* Valor por defecto */
    text-transform: uppercase;  /* MAYÚSCULAS */
    text-transform: lowercase;  /* minúsculas */
    text-transform: capitalize; /* Primera Letra En Mayúscula */
}
```

Text-Decoration: Decoración de Texto

```
.decoraciones {
    text-decoration: none;             /* Sin decoración */
    text-decoration: underline;        /* Subrayado */
    text-decoration: overline;         /* Línea arriba */
    text-decoration: line-through;     /* Tachado */
    text-decoration: underline overline; /* Múltiples */

    /* Control detallado (CSS3) */
    text-decoration-line: underline;
    text-decoration-color: red;
    text-decoration-style: wavy;
    text-decoration-thickness: 2px;
}
```

Letter-Spacing y Word-Spacing

```
.espaciado {
    letter-spacing: normal;   /* Valor por defecto */
    letter-spacing: 1px;      /* Espaciado entre letras */
    letter-spacing: -0.5px;   /* Espaciado negativo */

    word-spacing: normal;     /* Valor por defecto */
    word-spacing: 5px;        /* Espaciado entre palabras */
}
```

🎨 4. Colores en CSS

Formatos de Color

1. Nombres de Colores

```
.colores-nombres {
    color: red;
    color: blue;
    color: green;
    color: white;
    color: black;
    color: transparent;
}
```

2. Hexadecimal

```
.colores-hex {
    color: #ff0000;   /* Rojo completo */
    color: #00ff00;   /* Verde completo */
    color: #0000ff;   /* Azul completo */
    color: #fff;      /* Blanco (versión corta) */
    color: #000;      /* Negro (versión corta) */
    color: #f39c12;   /* Color personalizado */

    /* Con transparencia (CSS3) */
    color: #ff000080; /* Rojo 50% transparente */
}
```

3. RGB y RGBA

```
.colores-rgb {
    color: rgb(255, 0, 0);       /* Rojo */
    color: rgb(0, 255, 0);       /* Verde */
    color: rgb(0, 0, 255);       /* Azul */

    /* Con transparencia */
    color: rgba(255, 0, 0, 0.5); /* Rojo 50% transparente */
    color: rgba(0, 0, 0, 0.8);   /* Negro 80% opaco */
}
```

4. HSL y HSLA

```
.colores-hsl {
    /* HSL: Hue (matiz), Saturation (saturación), Lightness (luminosidad) */
    color: hsl(0, 100%, 50%);       /* Rojo */
    color: hsl(120, 100%, 50%);     /* Verde */
    color: hsl(240, 100%, 50%);     /* Azul */

    /* Con transparencia */
    color: hsla(0, 100%, 50%, 0.5); /* Rojo 50% transparente */
}
```

Ejemplo de Paleta de Colores

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Paletas de Colores</title>
    <style>
        body {
            font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, sans-serif;
            margin: 0;
            padding: 40px 20px;
            background-color: #f8f9fa;
        }

        .container {
            max-width: 1000px;
            margin: 0 auto;
            background: white;
            padding: 40px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        h1 {
            text-align: center;
            color: #1619FF;
            margin-bottom: 2rem;
        }

        .color-section {
            margin-bottom: 3rem;
        }

        .color-grid {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
            gap: 20px;
            margin-top: 20px;
        }

        .color-card {
            border-radius: 8px;
            overflow: hidden;
            box-shadow: 0 2px 8px rgba(0,0,0,0.1);
        }

        .color-preview {
            height: 120px;
            display: flex;
            align-items: center;
            justify-content: center;
            color: white;
            font-weight: bold;
            text-shadow: 1px 1px 2px rgba(0,0,0,0.3);
        }

        .color-info {
            padding: 15px;
            background: white;
        }

        .color-name {
            font-weight: bold;
            margin-bottom: 5px;
        }

        .color-code {
            font-family: monospace;
            font-size: 0.9em;
            color: #666;
        }

        /* Colores de MindHub */
        .primary-blue-mh { background-color: #1619FF; }
```

```
        .primary-fuchsia-mh { background-color: #FA01FA; }
        .primary-green-mh { background-color: #A9F00F; color: #000; }

        /* Escala de grises */
        .gray-100 { background-color: #f8f9fa; color: #333; }
        .gray-500 { background-color: #6c757d; }
        .gray-900 { background-color: #212529; }

        /* Demostración de uso */
        .demo-text {
            margin-top: 30px;
            padding: 20px;
            border-radius: 8px;
            border-left: 4px solid #1619FF;
            background-color: #ecf0f1;
        }

        .text-primary { color: #1619FF; }
        .text-secondary { color: #FA01FA; }
        .text-highlight { color: #A9F00F; }
    </style>
</head>
<body>
    <div class="container">
        <h1>Paletas de Colores en CSS</h1>

        <div class="color-section">
            <h2 style="color: #FA01FA;">Colores MindHub</h2>
            <div class="color-grid">
                <div class="color-card">
                    <div class="color-preview primary-blue-mh">Azul</div>
                    <div class="color-info">
                        <div class="color-name">Azul MindHub</div>
                        <div class="color-code">#1619FF</div>
                    </div>
                </div>

                <div class="color-card">
                    <div class="color-preview primary-fuchsia-mh">Fucsia</div>
                    <div class="color-info">
                        <div class="color-name">Fucsia MindHub</div>
                        <div class="color-code">#FA01FA</div>
                    </div>
                </div>

                <div class="color-card">
                    <div class="color-preview primary-green-mh">Verde</div>
                    <div class="color-info">
                        <div class="color-name">Verde MindHub</div>
                        <div class="color-code">#A9F00F</div>
                    </div>
                </div>
            </div>
        </div>

        <div class="demo-text">
            <h3>Aplicación Práctica de Colores</h3>
            <p>
                Aquí vemos diferentes aplicaciones de color en texto:
                <span class="text-primary">texto primario</span>,
                <span class="text-secondary">texto secundario</span>, y
                <span class="text-highlight" style="background-color: #333; padding: 2px;">texto resaltado</span>.
            </p>
        </div>
    </div>
</body>
</html>
```

✨ 5. Sombras y Efectos Visuales Básicos

Text-Shadow: Sombras de Texto

```
.sombras-texto {
    /* Sintaxis: offset-x offset-y blur-radius color */
    text-shadow: 2px 2px 4px rgba(0,0,0,0.3);

    /* Múltiples sombras */
    text-shadow:
        1px 1px 2px rgba(0,0,0,0.3),
        2px 2px 4px rgba(0,0,0,0.1);

    /* Efectos específicos */
    text-shadow: 0 1px 0 rgba(255,255,255,0.8); /* Grabado */
    text-shadow: 0 -1px 0 rgba(0,0,0,0.5);    /* Relieve */
    text-shadow: 2px 2px 0 #000;              /* Sombra sólida */
}
```

Box-Shadow: Sombras de Caja

```
.sombras-caja {
    /* Sintaxis: offset-x offset-y blur-radius spread-radius color */
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);

    /* Sombra interna */
    box-shadow: inset 0 1px 3px rgba(0,0,0,0.1);

    /* Múltiples sombras */
    box-shadow:
        0 1px 3px rgba(0,0,0,0.1),
        0 1px 2px rgba(0,0,0,0.06);

    /* Efectos específicos */
    box-shadow: 0 10px 25px rgba(0,0,0,0.1);     /* Elevación */
    box-shadow: 0 0 10px rgba(52,152,219,0.3); /* Brillo colorido */
}
```

Opacity: Transparencia

```
.transparencia {
    opacity: 1;   /* Completamente opaco (defecto) */
    opacity: 0.8; /* 80% opaco */
    opacity: 0.5; /* 50% opaco */
    opacity: 0;   /* Completamente transparente */
}
```

Ejemplo de Efectos Visuales

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Efectos Visuales CSS</title>
    <style>
        body {
            font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, sans-serif;
            margin: 0;
            padding: 40px 20px;
            background: linear-gradient(135deg, #1619FF 0%, #FA01FA 100%);
            min-height: 100vh;
        }

        .container {
            max-width: 1000px;
            margin: 0 auto;
        }

        h1 {
            text-align: center;
            color: white;
            font-size: 3rem;
            margin-bottom: 3rem;
            text-shadow: 0 2px 4px rgba(0,0,0,0.3);
        }

        .effects-grid {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
            gap: 30px;
            margin-bottom: 40px;
        }

        .effect-card {
            background: white;
            border-radius: 12px;
            padding: 30px;
            text-align: center;
            transition: transform 0.3s ease, box-shadow 0.3s ease;
        }

        .effect-card:hover {
            transform: translateY(-5px);
        }

        /* Efectos de sombra */
        .shadow-soft {
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        .shadow-medium {
            box-shadow: 0 4px 20px rgba(0,0,0,0.15);
        }

        .shadow-strong {
            box-shadow: 0 8px 30px rgba(0,0,0,0.2);
        }

        .shadow-colored {
            box-shadow: 0 4px 20px rgba(22, 25, 255, 0.3);
        }

        .shadow-inset {
            box-shadow: inset 0 2px 10px rgba(0,0,0,0.1);
            background-color: #f8f9fa;
        }

        /* Efectos de texto */
        .text-embossed {
            color: #333;
            text-shadow: 0 1px 0 rgba(255,255,255,0.8);
        }
```

```
.text-engraved {
    color: #666;
    text-shadow: 0 -1px 0 rgba(0,0,0,0.5);
}

.text-glow {
    color: #1619FF;
    text-shadow: 0 0 10px rgba(22, 25, 255, 0.5);
}

.text-3d {
    color: #e74c3c;
    text-shadow:
        1px 1px 0 #c0392b,
        2px 2px 0 #a93226,
        3px 3px 0 #922b21,
        4px 4px 6px rgba(0,0,0,0.3);
}

/* Efectos de transparencia */
.opacity-full { opacity: 1; }
.opacity-high { opacity: 0.8; }
.opacity-medium { opacity: 0.6; }
.opacity-low { opacity: 0.4; }

/* Efectos especiales */
.glass-effect {
    background: rgba(255,255,255,0.1);
    backdrop-filter: blur(10px);
    border: 1px solid rgba(255,255,255,0.2);
    color: white;
}

.neon-effect {
    background: #1a1a1a;
    color: #A9F00F;
    text-shadow:
        0 0 5px #A9F00F,
        0 0 10px #A9F00F,
        0 0 15px #A9F00F,
        0 0 20px #A9F00F;
    border: 1px solid #A9F00F;
    box-shadow: 0 0 10px rgba(169, 240, 15, 0.3);
}

.gradient-text {
    background: linear-gradient(45deg, #FA01FA, #A9F00F, #1619FF);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
    background-clip: text;
    font-weight: bold;
    font-size: 1.5rem;
}

.card-title {
    font-size: 1.2rem;
    font-weight: 600;
    margin-bottom: 15px;
    color: #2c3e50;
}

.card-description {
    color: #7f8c8d;
    line-height: 1.6;
}

/* Demostración de capas de transparencia */
.transparency-demo {
    background: white;
    border-radius: 12px;
    padding: 30px;
    margin-top: 30px;
    position: relative;
    overflow: hidden;
    min-height: 200px;
```

```
        }

        .layer {
            position: absolute;
            width: 100px;
            height: 100px;
            border-radius: 50%;
        }

        .layer-1 {
            background: rgba(250, 1, 250, 0.8);
            top: 20px;
            left: 20px;
        }

        .layer-2 {
            background: rgba(22, 25, 255, 0.8);
            top: 40px;
            left: 80px;
        }

        .layer-3 {
            background: rgba(169, 240, 15, 0.8);
            top: 60px;
            left: 140px;
        }

        .demo-content {
            position: relative;
            z-index: 10;
            background: rgba(255,255,255,0.9);
            padding: 20px;
            border-radius: 8px;
            margin-left: 200px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Efectos Visuales con CSS</h1>

        <div class="effects-grid">
            <!-- Sombras -->
            <div class="effect-card shadow-soft">
                <div class="card-title">Sombra Suave</div>
                <div class="card-description">Una sombra sutil para una leve profundidad.</div>
            </div>

            <!-- Efectos de texto -->
            <div class="effect-card">
                <div class="card-title text-glow">Texto Brillante</div>
                <div class="card-description">Efecto de resplandor alrededor del texto.</div>
            </div>

            <!-- Efectos especiales -->
            <div class="effect-card neon-effect">
                <div class="card-title">Efecto Neón</div>
                <div class="card-description">Efecto de luz neón vibrante.</div>
            </div>
        </div>

        <!-- Demostración de transparencias -->
        <div class="transparency-demo">
            <div class="layer layer-1"></div>
            <div class="layer layer-2"></div>
            <div class="layer layer-3"></div>

            <div class="demo-content">
                <h3>Capas de Transparencia</h3>
                <p>Las capas con transparencias se superponen creando nuevos colores.</p>
            </div>
        </div>
    </div>
</body>
</html>
```

🏆 6. Mejores Prácticas de Tipografía y Color

Sistema de Diseño Tipográfico

```
/* Variables CSS para consistencia */
:root {
    /* Escala tipográfica */
    --font-size-xs: 0.75rem;   /* 12px */
    --font-size-sm: 0.875rem;  /* 14px */
    --font-size-base: 1rem;    /* 16px */
    --font-size-lg: 1.125rem;  /* 18px */
    --font-size-xl: 1.25rem;   /* 20px */
    --font-size-2xl: 1.5rem;   /* 24px */

    /* Pesos de fuente */
    --font-weight-light: 300;
    --font-weight-normal: 400;
    --font-weight-bold: 700;

    /* Interlineado */
    --line-height-normal: 1.5;
    --line-height-relaxed: 1.75;
}

/* Clases utilitarias */
.text-xs { font-size: var(--font-size-xs); }
.text-base { font-size: var(--font-size-base); }
.text-2xl { font-size: var(--font-size-2xl); }
```

Paleta de Colores Sistemática

```
:root {
    /* Colores MindHub */
    --color-primary: #1619FF;
    --color-secondary: #FA01FA;
    --color-highlight: #A9F00F;

    /* Grises */
    --color-gray-100: #f3f4f6;
    --color-gray-500: #6b7280;
    --color-gray-900: #111827;

    /* Colores semánticos */
    --color-success: #10b981;
    --color-error: #ef4444;
}
```

Accesibilidad y Legibilidad

```
/* Contraste mínimo para accesibilidad */
.text-on-light {
    color: var(--color-gray-900);
}

.text-on-dark {
    color: var(--color-gray-100);
}

/* Tamaños mínimos para lectura */
body {
    font-size: 16px;       /* Nunca menor a 16px */
    line-height: 1.6;      /* Mínimo 1.5 para legibilidad */
}

/* Responsive typography */
@media (max-width: 768px) {
    html {
        font-size: 14px;   /* Base más pequeña en móvil */
    }

    h1 { font-size: 2rem; }
    h2 { font-size: 1.5rem; }
}
```

🛠️ 7. Herramientas y Recursos

Herramientas de Color

- **Adobe Color**: Generador de paletas
- **Coolors.co**: Paletas de colores
- **Contrast Checker**: Verificar accesibilidad
- **Color Hunt**: Inspiración de colores

Herramientas de Tipografía

- **Google Fonts**: Fuentes web gratuitas
- **Font Pair**: Combinaciones de fuentes
- **Type Scale**: Calculadora de escalas tipográficas
- **Modular Scale**: Proporciones armoniosas

Testing y Accesibilidad

- **WAVE**: Evaluador de accesibilidad web
- **Colour Contrast Analyser**: Contraste de colores
- **Lighthouse**: Auditoría de rendimiento y accesibilidad

6. Gestión de Assets

📁 1. Introducción a la Gestión de Assets

¿Qué son los Assets?

Los **assets** (recursos) son todos los archivos externos que utiliza nuestro sitio web:

- **Hojas de estilo CSS**
- **Imágenes** (JPG, PNG, SVG, WebP)
- **Fuentes** (WOFF2, WOFF, TTF)
- **Íconos** (SVG, PNG, ICO)
- **Videos y audio**
- **Scripts JavaScript**

¿Por qué es importante gestionarlos bien?

```
BENEFICIOS DE BUENA GESTIÓN
├── RENDIMIENTO
│   ├── Carga más rápida
│   ├── Menor uso de ancho de banda
│   └── Mejor experiencia de usuario
├── MANTENIBILIDAD
│   ├── Código más organizado
│   ├── Fácil localización de archivos
│   └── Colaboración en equipo
├── ESCALABILIDAD
│   ├── Proyecto puede crecer ordenadamente
│   ├── Reutilización de componentes
│   └── Actualizaciones eficientes
└── SEO Y ACCESIBILIDAD
    ├── Mejores tiempos de carga
    └── Accesibilidad mejorada
```

↔️ 2. Rutas: Absolutas vs Relativas

Tipos de Rutas

1. Rutas Absolutas

```
/* Ruta absoluta desde la raíz del servidor */
background-image: url('/assets/images/background.jpg');

/* Ruta absoluta completa (URL externa) */
background-image: url('https://example.com/images/background.jpg');

/* En HTML */
<link rel="stylesheet" href="/css/styles.css">
<img src="/images/logo.png" alt="Logo">
```

2. Rutas Relativas

```
/* Relativa al archivo CSS actual */
background-image: url('../images/background.jpg');
background-image: url('./icons/arrow.svg');
background-image: url('background.jpg'); /* misma carpeta */

/* En HTML */
<link rel="stylesheet" href="css/styles.css">
<link rel="stylesheet" href="../css/styles.css">
<img src="images/logo.png" alt="Logo">
```

Estructura de Carpetas Ejemplo

```
proyecto-web/
├── index.html
├── css/
│   ├── styles.css
│   ├── components/
│   │   ├── header.css
│   │   ├── footer.css
│   │   └── buttons.css
│   └── utilities/
│       ├── reset.css
│       └── variables.css
├── images/
│   ├── backgrounds/
│   │   ├── hero-bg.jpg
│   │   └── section-bg.png
│   ├── icons/
│   │   ├── social/
│   │   │   ├── facebook.svg
│   │   │   └── twitter.svg
│   │   └── ui/
│   │       ├── arrow.svg
│   │       └── close.svg
│   └── content/
│       ├── product1.jpg
│       └── product2.jpg
├── fonts/
│   ├── Roboto-Regular.woff2
│   ├── Roboto-Bold.woff2
│   └── icons.woff2
└── js/
    └── main.js
```

Ejemplos Prácticos de Rutas

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="/css/global.css">
</head>
<body>
    <!-- Rutas relativas para imágenes -->
    <img src="/images/icons/search.svg" alt="Buscar">
</body>
</html>
```

```css
/* En css/styles.css */

/* Ruta relativa: subir un nivel, entrar a images */
.hero {
    background-image: url('../images/backgrounds/hero-bg.jpg');
}

/* Ruta relativa: mismo nivel que CSS, entrar a icons */
.icon-arrow {
    background-image: url('../images/icons/ui/arrow.svg');
}

/* Ruta absoluta desde raíz del servidor */
.header-bg {
    background-image: url('/images/backgrounds/header-bg.png');
}

/* Fuentes con rutas relativas */
@font-face {
    font-family: 'Roboto';
    font-weight: 400;
}
```

🗂️ 3. Organización de Archivos CSS

Metodologías de Organización

1. Arquitectura por Componentes

```
css/
├── base/
│   ├── reset.css         /* Normalización */
│   ├── typography.css    /* Tipografía base */
│   └── variables.css     /* Variables CSS */
├── components/
│   ├── buttons.css       /* Componente botones */
│   ├── cards.css         /* Componente tarjetas */
│   ├── navbar.css        /* Navegación */
│   ├── modal.css         /* Modales */
│   └── forms.css         /* Formularios */
├── layout/
│   ├── header.css        /* Encabezado */
│   ├── footer.css        /* Pie de página */
│   ├── sidebar.css       /* Barra lateral */
│   └── grid.css          /* Sistema de grid */
├── pages/
│   ├── home.css          /* Página inicio */
│   ├── about.css         /* Página acerca */
│   └── contact.css       /* Página contacto */
├── utilities/
│   ├── helpers.css       /* Clases utilitarias */
│   ├── responsive.css    /* Media queries */
│   └── animations.css    /* Animaciones */
└── main.css              /* Archivo principal */
```

2. Metodología SMACSS (Scalable and Modular Architecture)

```css
/* main.css - Archivo principal que importa todo */

/* 1. Base - Elementos base sin clases */
@import 'base/reset.css';
@import 'base/typography.css';

/* 2. Layout - Componentes de diseño principal */
@import 'layout/header.css';
@import 'layout/navigation.css';
@import 'layout/sidebar.css';
@import 'layout/footer.css';

/* 3. Modules - Componentes reutilizables */
@import 'modules/buttons.css';
@import 'modules/cards.css';
@import 'modules/forms.css';

/* 4. State - Estados de los módulos */
@import 'state/active.css';
@import 'state/hidden.css';

/* 5. Theme - Colores y efectos visuales */
@import 'theme/colors.css';
@import 'theme/effects.css';
```

Ejemplo de Estructura Modular

```css
@import 'base/reset.css';
@import 'base/typography.css';
@import 'layout/header.css';
@import 'modules/buttons.css';
@import 'state/active.css';
@import 'theme/colors.css';
```

🖼️ 4. Optimización de Imágenes para Web

Formatos de Imagen

1. JPG/JPEG

```css
/* Ideal para: fotografías, imágenes con muchos colores */
.photo-background {
    background-image: url('../images/photography/landscape.jpg');
    /* Buena compresión, pérdida de calidad aceptable */
}
```

- ✅ Excelente compresión para fotografías
- ✅ Amplio soporte en navegadores
- ❌ No soporta transparencia
- ❌ Pérdida de calidad al comprimir

2. PNG

```css
/* Ideal para: logos, íconos, imágenes con transparencia */
.logo {
    background-image: url('../images/logos/company-logo.png');
    /* Sin pérdida de calidad, soporta transparencia */
}
```

- ✅ Sin pérdida de calidad
- ✅ Soporta transparencia
- ✅ Ideal para gráficos simples
- ❌ Archivos más pesados

3. SVG

```
/* Ideal para: íconos, ilustraciones simples, logos escalables */
.icon {
    background-image: url('../images/icons/arrow.svg');
    /* Escalable sin pérdida de calidad */
}

/* SVG inline en CSS */
.icon-inline {
    background-image: url('data:image/svg+xml,<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"><path d="M12 2l3.09 6.26L22 9.27l-5 4.87 1.18 6.88L12 17.77l-6.18 3.25L7 14.14 2 9.27l6.91-1.01L12 2z" fill="currentColor"/></svg>');
}
```

- ✅ Escalable infinitamente
- ✅ Archivos muy pequeños
- ✅ Editable con CSS y JavaScript
- ✅ Perfecto para íconos y logos
- ❌ No ideal para fotografías complejas

4. WebP (Moderno)

```
/* Formato moderno con mejor compresión */
.modern-image {
    /* Fallback para navegadores que no soportan WebP */
    background-image: url('../images/hero-image.webp');
}
```

- ✅ Mejor compresión que JPG y PNG
- ✅ Soporta transparencia y animación
- ❌ Soporte limitado en navegadores antiguos

Técnicas de Optimización

1. Responsive Images con CSS

```
/* Imagen que se adapta al contenedor */
.responsive-image {
    width: 100%;
    max-width: 800px;
}

/* Background images responsivas */
.hero-section {
    background-image: url('../images/hero-mobile.jpg');
    min-height: 400px;
}

@media (min-width: 768px) {
    .hero-section {
        background-image: url('../images/hero-tablet.jpg');
    }
}

@media (min-width: 1200px) {
    .hero-section {
        background-image: url('../images/hero-desktop.jpg');
    }
}
```

2. Lazy Loading con CSS

```css
/* Imagen placeholder mientras carga */
.lazy-image {
    background-color: #f0f0f0;
    transition: opacity 0.3s ease;
}

.lazy-image.loaded {
    background-image: url('../images/actual-image.jpg');
}
```

🔡 5. Gestión de Fuentes Web

Carga de Fuentes Optimizada

1. Google Fonts

```css
/* En HTML - Método más simple */
/* <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap" rel="stylesheet"> */

/* En CSS - @import (menos eficiente) */
@import url('https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap');

/* Uso de la fuente */
body {
    font-family: 'Roboto', -apple-system, BlinkMacSystemFont, 'Segoe UI', sans-serif;
}
```

2. Fuentes Auto-hospedadas

```css
/* Definición de fuente personalizada */
@font-face {
    font-family: 'MiFuentePersonalizada';
    src: url('../fonts/MiFuente-Regular.woff2') format('woff2'),
         url('../fonts/MiFuente-Regular.woff') format('woff');
    font-weight: 400;
    font-style: normal;
    font-display: swap; /* Mejora rendimiento */
}

/* Múltiples pesos de la misma fuente */
@font-face {
    font-family: 'MiFuentePersonalizada';
    src: url('../fonts/MiFuente-Bold.woff2') format('woff2'),
         url('../fonts/MiFuente-Bold.woff') format('woff');
    font-weight: 700;
    font-style: normal;
    font-display: swap;
}

/* Uso */
.titulo {
    font-family: 'MiFuentePersonalizada', serif;
    font-weight: 700;
}
```

3. Font-Display para Rendimiento

```
@font-face {
    font-family: 'MiFuente';
    src: url('../fonts/MiFuente.woff2') format('woff2');
    font-display: swap; /* Opciones: auto, block, swap, fallback, optional */
}

/* font-display valores:
- auto: comportamiento por defecto del navegador
- block: bloquea hasta que carge la fuente (puede causar FOIT)
- swap: muestra fuente fallback inmediatamente, cambia cuando carga
- fallback: breve período de bloqueo, luego fallback
- optional: usa fuente solo si carga rápidamente
*/
```

🛠️ 6. Herramientas de Optimización Básicas

Herramientas Online Gratuitas

1. Optimización de Imágenes

- **TinyPNG/TinyJPG**: Compresión automática con pérdida mínima de calidad
- **Squoosh**: Herramienta web de Google para comparar formatos y compresión
- **ImageOptim**: Compresión sin pérdida de calidad
- **SVGOMG**: Optimización de archivos SVG

2. Validación y Testing

- **CSS Validator (W3C)**: Validar sintaxis CSS
- **PageSpeed Insights**: Analizar rendimiento de carga
- **GTmetrix**: Métricas de rendimiento web
- **Lighthouse**: Auditoría de rendimiento en DevTools

3. Generadores y Utilidades

- **CSS Gradient Generator**: Crear degradados visualmente
- **Google Fonts**: Seleccionar y optimizar fuentes web
- **Favicon Generator**: Crear íconos en múltiples formatos
- **Can I Use**: Verificar compatibilidad de navegadores

Consejos de Optimización Manual

Compresión de Imágenes

```css
/* Ejemplo de optimización manual de imágenes: */

/* ❌ Malo - imagen muy pesada */
.hero-bad {
    background-image: url('../images/hero-photo-5mb.jpg'); /* 5MB */
}

/* ✅ Bueno - imagen optimizada */
.hero-good {
    background-image: url('../images/hero-photo-compressed.jpg'); /* 150KB */
    background-size: cover;
    background-position: center;
}

/* ✅ Mejor - con fallback y responsive */
.hero-best {
    background-image: url('../images/hero-mobile.jpg'); /* Para móviles */
    background-size: cover;
    background-position: center;
}

@media (min-width: 768px) {
    .hero-best {
        background-image: url('../images/hero-tablet.jpg'); /* Para tablets */
    }
}

@media (min-width: 1200px) {
    .hero-best {
        background-image: url('../images/hero-desktop.jpg'); /* Para desktop */
    }
}
```

Organización Manual de CSS

```css
/* Organización eficiente sin herramientas complejas */

/* 1. Variables CSS nativas */
:root {
    --color-primary: #1619FF;
    --color-secondary: #6c757d;
    --color-success: #28a745;
    --spacing-base: 1rem;
    --font-family-base: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
}

/* 2. Reset básico */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* 3. Base */
body {
    font-family: var(--font-family-base);
    line-height: 1.6;
    color: #333;
}

/* 4. Componentes */
.btn {
    display: inline-block;
    padding: calc(var(--spacing-base) * 0.5) var(--spacing-base);
    background-color: var(--color-primary);
    color: white;
    text-decoration: none;
    border-radius: 4px;
    transition: background-color 0.3s ease;
}

.btn:hover {
    background-color: #0d0f9a;
}

/* 5. Utilidades */
.text-center { text-align: center; }
.mb-1 { margin-bottom: var(--spacing-base); }
.p-2 { padding: calc(var(--spacing-base) * 2); }
```

7. ¿Qué es Responsive Web Design?

📱 1. ¿Qué es Responsive Web Design?

Definición

Responsive Web Design (RWD) es un enfoque de diseño web que hace que las páginas web se vean bien en todos los dispositivos (desktop, tablets, móviles), adaptándose automáticamente al tamaño de pantalla del usuario.

Historia y Evolución

```
EVOLUCIÓN DEL DISEÑO WEB
├── ERA FIJA (1990s-2000s)
│   ├── Diseños de ancho fijo (960px, 1024px)
│   ├── Solo para computadoras de escritorio
│   └── Layouts con tablas
├── ERA RESPONSIVE (2010+)
│   ├── Ethan Marcotte introduce el término (2010)
│   ├── Auge de smartphones y tablets
│   ├── CSS3 Media Queries
│   └── Grids fluidos
└── ERA MOBILE-FIRST (2012+)
    ├── Más tráfico móvil que desktop
    ├── Progressive enhancement
    ├── Performance-focused
    └── Google Mobile-First Indexing
```

¿Por qué es importante?

Estadísticas Actuales (2025)

- 📱 **60%+** del tráfico web es móvil
- 🚀 **53%** de usuarios abandonan sitios que tardan más de 3 segundos en cargar
- 🔍 **Google** usa mobile-first indexing desde 2019
- 💰 **E-commerce** móvil representa 70%+ de las ventas online

Beneficios del Responsive Design

```
/* Un solo código base para todos los dispositivos */
.responsive-benefits {
    /* ✅ Mantenimiento más fácil */
    /* ✅ Mejor SEO (una sola URL) */
    /* ✅ Menor costo de desarrollo */
    /* ✅ Mejor experiencia de usuario */
    /* ✅ Future-proof para nuevos dispositivos */
}
```

🏛️ 2. Los Pilares del Responsive Design

1. Grids Fluidos

En lugar de layouts fijos, usamos porcentajes y unidades relativas:

```
/* ❌ Diseño fijo - no responsive */
.container-fixed {
    width: 960px;       /* Fijo en píxeles */
    margin: 0 auto;
}

.sidebar-fixed {
    width: 300px;       /* Ancho fijo */
    float: left;
}

.content-fixed {
    width: 660px;       /* Ancho fijo */
    float: right;
}

/* ✅ Diseño fluido - responsive */
.container-fluid {
    max-width: 1200px;  /* Máximo, pero se adapta */
    width: 90%;         /* Porcentaje del viewport */
    margin: 0 auto;
}

.sidebar-fluid {
    width: 30%;         /* Porcentaje del contenedor */
    float: left;
}

.content-fluid {
    width: 70%;         /* Porcentaje del contenedor */
    float: right;
}
```

2. Imágenes Flexibles

Las imágenes deben adaptarse al contenedor:

```
/* Imágenes responsive básicas */
.responsive-image {
    max-width: 100%;    /* Nunca más grande que su contenedor */
    height: auto;         /* Mantiene la proporción */
}

/* Para imágenes de fondo */
.responsive-background {
    background-size: cover; /* Cubre todo el área */
    background-position: center;
    background-repeat: no-repeat;
}
```

3. Media Queries

Aplicar estilos específicos según el dispositivo:

```css
/* Estilos base (mobile-first) */
.navigation {
    display: block;
}

/* Tablet y superior */
@media (min-width: 768px) {
    .navigation {
        display: flex;
    }
}

/* Desktop */
@media (min-width: 1024px) {
    .navigation {
        justify-content: space-between;
    }
}
```

📲 3. El Enfoque Mobile-First

¿Qué es Mobile-First?

Mobile-First es una estrategia de diseño que comienza por el diseño móvil y luego se expande hacia pantallas más grandes.

¿Por qué Mobile-First?

Ventajas Técnicas

```css
/* ✅ Mobile-First: Progresive Enhancement */
.element {
    /* Estilos base para móvil (más simples) */
    padding: 10px;
    display: block;
}

@media (min-width: 768px) {
    .element {
        padding: 20px;
        display: flex;
    }
}

/* ❌ Desktop-First: Graceful Degradation */
.element-desktop-first {
    /* Estilos complejos para desktop */
    padding: 40px;
    display: grid;
    transform: scale(1.1);
}

@media (max-width: 767px) {
    .element-desktop-first {
        padding: 10px;
        display: block;
        transform: none;
    }
}
```

Ventajas de Rendimiento

- **Menos CSS** se carga en móviles
- **Menos recursos** computacionales
- **Carga más rápida** en conexiones lentas
- **Menos JavaScript** complejo inicialmente

Principios del Mobile-First Design

1. Contenido Prioritario

```
<!-- Estructura que prioriza lo esencial -->
<header>
    <h1>Logo</h1>
    <button class="menu-toggle">☰</button>
</header>

<main>
    <section class="hero">
        <h2>Título Principal</h2>
        <p>Contenido esencial visible inmediatamente</p>
    </section>
</main>

<nav class="navigation hidden-mobile">
    <!-- Navegación secundaria oculta en móvil -->
</nav>
```

2. Navegación Adaptativa

```
/* Navegación mobile-first */
.main-nav {
    /* Oculta por defecto en móvil */
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100vh;
    background: rgba(0, 0, 0, 0.9);
    z-index: 1000;
}

.main-nav.active {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

.menu-toggle {
    display: block;
    background: none;
    border: none;
    font-size: 1.5rem;
    cursor: pointer;
}

/* Tablet y superior */
@media (min-width: 768px) {
    .main-nav {
        display: flex;
        position: static;
        width: auto;
        height: auto;
        background: transparent;
        flex-direction: row;
    }

    .menu-toggle {
        display: none;
    }
}
```

🌐 4. El Viewport Meta Tag

¿Qué es el Viewport?

El **viewport** es el área visible de una página web en el dispositivo del usuario. Sin configuración, los navegadores móviles intentan mostrar la página completa, haciéndola muy pequeña.

Configuración Básica

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Página Responsive</title>
</head>
```

Opciones del Viewport

```
<!-- Configuración básica recomendada -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- Opciones adicionales -->
<meta name="viewport" content="
    width=device-width,
    initial-scale=1.0,
    maximum-scale=5.0,
    minimum-scale=1.0,
    user-scalable=yes
">
```

Parámetros Explicados

- `width=device-width`: Ancho igual al dispositivo
- `initial-scale=1.0`: Zoom inicial 100%
- `maximum-scale=5.0`: Zoom máximo permitido
- `minimum-scale=1.0`: Zoom mínimo permitido
- `user-scalable=yes`: Permite zoom manual

Ejemplo Comparativo

❌ Sin Viewport Meta Tag

La página se ve muy pequeña en móviles. El usuario debe hacer zoom para leer el contenido.

✅ Con Viewport Meta Tag

La página se adapta perfectamente al ancho del dispositivo. Texto legible sin zoom.

📄 5. Estrategias de Contenido Mobile-First

Jerarquía de Contenido

```
<!-- Estructura mobile-first -->
<article class="post">
    <!-- 1. Lo más importante primero -->
    <header class="post-header">
        <h1 class="post-title">Título del Artículo</h1>
        <p class="post-subtitle">Subtítulo descriptivo</p>
    </header>

    <!-- 2. Lead o introducción -->
    <p class="lead">Párrafo principal que engancha al usuario...</p>

    <!-- 3. Contenido principal -->
    <div class="post-content">
        <p>Contenido del artículo...</p>
    </div>

    <!-- 4. Información secundaria (oculta en móvil) -->
    <aside class="post-meta hidden-mobile">
        <p>Información adicional, tags, autor, fecha...</p>
    </aside>

    <!-- 5. Call-to-action prominente en móvil -->
    <div class="cta mobile-prominent">
        <button>Acción Principal</button>
    </div>
</article>
```

CSS para Jerarquía

```
/* Estilos base mobile-first */
.post {
    padding: 15px;
    margin-bottom: 20px;
}

.post-title {
    font-size: 1.5rem;
    line-height: 1.2;
    margin-bottom: 0.5rem;
    color: #2c3e50;
}

.post-subtitle {
    font-size: 1rem;
    color: #6c757d;
    margin-bottom: 1rem;
}

.lead {
    font-size: 1.1rem;
    font-weight: 500;
    line-height: 1.4;
    margin-bottom: 1.5rem;
}

/* Ocultar información secundaria en móvil */
.hidden-mobile {
    display: none;
}

.mobile-prominent {
    width: 100%;
    text-align: center;
    margin-top: 2rem;
}

.mobile-prominent button {
    width: 100%;
    padding: 15px;
    font-size: 1.1rem;
    background-color: #1619FF;
    color: white;
    border: none;
    border-radius: 5px;
}

/* Tablet y superior */
@media (min-width: 768px) {
    .post {
        padding: 30px;
        display: grid;
        grid-template-columns: 2fr 1fr;
        gap: 30px;
    }

    .post-title {
        font-size: 2rem;
    }

    .hidden-mobile {
        display: block;
    }

    .mobile-prominent {
        width: auto;
        text-align: left;
    }

    .mobile-prominent button {
        width: auto;
        padding: 10px 20px;
        font-size: 1rem;
```

```
    }
}
```

Navegación Progressive Enhancement

```css
/* Navegación mobile-first */
.main-navigation {
    background: #333;
    color: white;
    padding: 1rem;
}

.nav-toggle {
    display: block;
    background: none;
    border: none;
    color: white;
    font-size: 1.5rem;
    cursor: pointer;
    float: right;
}

.nav-menu {
    display: none;
    list-style: none;
    padding: 0;
    margin: 1rem 0 0 0;
}

.nav-menu.active {
    display: block;
}

.nav-menu li {
    border-bottom: 1px solid #555;
    padding: 0;
}

.nav-menu a {
    display: block;
    padding: 15px 0;
    color: white;
    text-decoration: none;
    transition: background-color 0.3s ease;
}

.nav-menu a:hover {
    background-color: #555;
    padding-left: 10px;
}

/* Enhanced para pantallas más grandes */
@media (min-width: 768px) {
    .nav-toggle {
        display: none;
    }

    .nav-menu {
        display: flex;
        margin: 0;
        justify-content: space-around;
    }

    .nav-menu li {
        border-bottom: none;
        flex: 1;
        text-align: center;
    }

    .nav-menu a {
        padding: 15px 10px;
    }

    .nav-menu a:hover {
        padding-left: 10px;
        background-color: #1619FF;
```

```
    }
}
```

🧪 6. Herramientas de Testing

DevTools del Navegador

```
// Simular diferentes dispositivos en DevTools
// Chrome/Firefox/Safari: F12 → Device Toolbar

// Dispositivos comunes para testing:
// - iPhone SE (375x667)
// - iPhone 12 Pro (390x844)
// - iPad (768x1024)
// - Desktop (1920x1080)

// Verificar viewport actual
console.log('Viewport:', window.innerWidth + 'x' + window.innerHeight);
console.log('Device Pixel Ratio:', window.devicePixelRatio);
```

7.1. ¿Qué son las Media Queries?

📱 1. ¿Qué es Responsive Web Design?

Definición

Responsive Web Design (RWD) es un enfoque de diseño web que hace que las páginas web se vean bien en todos los dispositivos (desktop, tablets, móviles), adaptándose automáticamente al tamaño de pantalla del usuario.

Historia y Evolución

```
EVOLUCIÓN DEL DISEÑO WEB
├── ERA FIJA (1990s-2000s)
│   ├── Diseños de ancho fijo (960px, 1024px)
│   ├── Solo para computadoras de escritorio
│   └── Layouts con tablas
├── ERA RESPONSIVE (2010+)
│   ├── Ethan Marcotte introduce el término (2010)
│   ├── Auge de smartphones y tablets
│   ├── CSS3 Media Queries
│   └── Grids fluidos
└── ERA MOBILE-FIRST (2012+)
    ├── Más tráfico móvil que desktop
    ├── Progressive enhancement
    ├── Performance-focused
    └── Google Mobile-First Indexing
```

¿Por qué es importante?

Estadísticas Actuales (2025)

- 📱 **60%+** del tráfico web es móvil
- 🚀 **53%** de usuarios abandonan sitios que tardan más de 3 segundos en cargar
- 🔍 **Google** usa mobile-first indexing desde 2019
- 💰 **E-commerce** móvil representa 70%+ de las ventas online

Beneficios del Responsive Design

```
/* Un solo código base para todos los dispositivos */
.responsive-benefits {
    /* ✅ Mantenimiento más fácil */
    /* ✅ Mejor SEO (una sola URL) */
    /* ✅ Menor costo de desarrollo */
    /* ✅ Mejor experiencia de usuario */
    /* ✅ Future-proof para nuevos dispositivos */
}
```

🏛️ 2. Los Pilares del Responsive Design

1. Grids Fluidos

En lugar de layouts fijos, usamos porcentajes y unidades relativas:

```
/* ❌ Diseño fijo - no responsive */
.container-fixed {
    width: 960px;       /* Fijo en píxeles */
    margin: 0 auto;
}

.sidebar-fixed {
    width: 300px;       /* Ancho fijo */
    float: left;
}

.content-fixed {
    width: 660px;       /* Ancho fijo */
    float: right;
}

/* ✅ Diseño fluido - responsive */
.container-fluid {
    max-width: 1200px;  /* Máximo, pero se adapta */
    width: 90%;         /* Porcentaje del viewport */
    margin: 0 auto;
}

.sidebar-fluid {
    width: 30%;         /* Porcentaje del contenedor */
    float: left;
}

.content-fluid {
    width: 70%;         /* Porcentaje del contenedor */
    float: right;
}
```

2. Imágenes Flexibles

Las imágenes deben adaptarse al contenedor:

```
/* Imágenes responsive básicas */
.responsive-image {
    max-width: 100%;    /* Nunca más grande que su contenedor */
    height: auto;         /* Mantiene la proporción */
}

/* Para imágenes de fondo */
.responsive-background {
    background-size: cover; /* Cubre todo el área */
    background-position: center;
    background-repeat: no-repeat;
}
```

3. Media Queries

Aplicar estilos específicos según el dispositivo:

```css
/* Estilos base (mobile-first) */
.navigation {
    display: block;
}

/* Tablet y superior */
@media (min-width: 768px) {
    .navigation {
        display: flex;
    }
}

/* Desktop */
@media (min-width: 1024px) {
    .navigation {
        justify-content: space-between;
    }
}
```

📲 3. El Enfoque Mobile-First

¿Qué es Mobile-First?

Mobile-First es una estrategia de diseño que comienza por el diseño móvil y luego se expande hacia pantallas más grandes.

¿Por qué Mobile-First?

Ventajas Técnicas

```css
/* ✅ Mobile-First: Progresive Enhancement */
.element {
    /* Estilos base para móvil (más simples) */
    padding: 10px;
    display: block;
}

@media (min-width: 768px) {
    .element {
        padding: 20px;
        display: flex;
    }
}

/* ❌ Desktop-First: Graceful Degradation */
.element-desktop-first {
    /* Estilos complejos para desktop */
    padding: 40px;
    display: grid;
    transform: scale(1.1);
}

@media (max-width: 767px) {
    .element-desktop-first {
        padding: 10px;
        display: block;
        transform: none;
    }
}
```

Ventajas de Rendimiento

- **Menos CSS** se carga en móviles
- **Menos recursos** computacionales
- **Carga más rápida** en conexiones lentas
- **Menos JavaScript** complejo inicialmente

Principios del Mobile-First Design

1. Contenido Prioritario

```
<!-- Estructura que prioriza lo esencial -->
<header>
    <h1>Logo</h1>
    <button class="menu-toggle">☰</button>
</header>

<main>
    <section class="hero">
        <h2>Título Principal</h2>
        <p>Contenido esencial visible inmediatamente</p>
    </section>
</main>

<nav class="navigation hidden-mobile">
    <!-- Navegación secundaria oculta en móvil -->
</nav>
```

2. Navegación Adaptativa

```
/* Navegación mobile-first */
.main-nav {
    /* Oculta por defecto en móvil */
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100vh;
    background: rgba(0, 0, 0, 0.9);
    z-index: 1000;
}

.main-nav.active {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

.menu-toggle {
    display: block;
    background: none;
    border: none;
    font-size: 1.5rem;
    cursor: pointer;
}

/* Tablet y superior */
@media (min-width: 768px) {
    .main-nav {
        display: flex;
        position: static;
        width: auto;
        height: auto;
        background: transparent;
        flex-direction: row;
    }

    .menu-toggle {
        display: none;
    }
}
```

🌐 4. El Viewport Meta Tag

¿Qué es el Viewport?

El **viewport** es el área visible de una página web en el dispositivo del usuario. Sin configuración, los navegadores móviles intentan mostrar la página completa, haciéndola muy pequeña.

Configuración Básica

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Página Responsive</title>
</head>
```

Opciones del Viewport

```
<!-- Configuración básica recomendada -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- Opciones adicionales -->
<meta name="viewport" content="
    width=device-width,
    initial-scale=1.0,
    maximum-scale=5.0,
    minimum-scale=1.0,
    user-scalable=yes
">
```

Parámetros Explicados

- **`width=device-width`** : Ancho igual al dispositivo
- **`initial-scale=1.0`** : Zoom inicial 100%
- **`maximum-scale=5.0`** : Zoom máximo permitido
- **`minimum-scale=1.0`** : Zoom mínimo permitido
- **`user-scalable=yes`** : Permite zoom manual

Ejemplo Comparativo

❌ Sin Viewport Meta Tag

La página se ve muy pequeña en móviles. El usuario debe hacer zoom para leer el contenido.

✅ Con Viewport Meta Tag

La página se adapta perfectamente al ancho del dispositivo. Texto legible sin zoom.

📄 5. Estrategias de Contenido Mobile-First

Jerarquía de Contenido

```
<!-- Estructura mobile-first -->
<article class="post">
    <!-- 1. Lo más importante primero -->
    <header class="post-header">
        <h1 class="post-title">Título del Artículo</h1>
        <p class="post-subtitle">Subtítulo descriptivo</p>
    </header>

    <!-- 2. Lead o introducción -->
    <p class="lead">Párrafo principal que engancha al usuario...</p>

    <!-- 3. Contenido principal -->
    <div class="post-content">
        <p>Contenido del artículo...</p>
    </div>

    <!-- 4. Información secundaria (oculta en móvil) -->
    <aside class="post-meta hidden-mobile">
        <p>Información adicional, tags, autor, fecha...</p>
    </aside>

    <!-- 5. Call-to-action prominente en móvil -->
    <div class="cta mobile-prominent">
        <button>Acción Principal</button>
    </div>
</article>
```

CSS para Jerarquía

```
/* Estilos base mobile-first */
.post {
    padding: 15px;
    margin-bottom: 20px;
}

.post-title {
    font-size: 1.5rem;
    line-height: 1.2;
    margin-bottom: 0.5rem;
    color: #2c3e50;
}

.post-subtitle {
    font-size: 1rem;
    color: #6c757d;
    margin-bottom: 1rem;
}

.lead {
    font-size: 1.1rem;
    font-weight: 500;
    line-height: 1.4;
    margin-bottom: 1.5rem;
}

/* Ocultar información secundaria en móvil */
.hidden-mobile {
    display: none;
}

.mobile-prominent {
    width: 100%;
    text-align: center;
    margin-top: 2rem;
}

.mobile-prominent button {
    width: 100%;
    padding: 15px;
    font-size: 1.1rem;
    background-color: #1619FF;
    color: white;
    border: none;
    border-radius: 5px;
}

/* Tablet y superior */
@media (min-width: 768px) {
    .post {
        padding: 30px;
        display: grid;
        grid-template-columns: 2fr 1fr;
        gap: 30px;
    }

    .post-title {
        font-size: 2rem;
    }

    .hidden-mobile {
        display: block;
    }

    .mobile-prominent {
        width: auto;
        text-align: left;
    }

    .mobile-prominent button {
        width: auto;
        padding: 10px 20px;
        font-size: 1rem;
```

```
    }
}
```

Navegación Progressive Enhancement

```css
/* Navegación mobile-first */
.main-navigation {
    background: #333;
    color: white;
    padding: 1rem;
}

.nav-toggle {
    display: block;
    background: none;
    border: none;
    color: white;
    font-size: 1.5rem;
    cursor: pointer;
    float: right;
}

.nav-menu {
    display: none;
    list-style: none;
    padding: 0;
    margin: 1rem 0 0 0;
}

.nav-menu.active {
    display: block;
}

.nav-menu li {
    border-bottom: 1px solid #555;
    padding: 0;
}

.nav-menu a {
    display: block;
    padding: 15px 0;
    color: white;
    text-decoration: none;
    transition: background-color 0.3s ease;
}

.nav-menu a:hover {
    background-color: #555;
    padding-left: 10px;
}

/* Enhanced para pantallas más grandes */
@media (min-width: 768px) {
    .nav-toggle {
        display: none;
    }

    .nav-menu {
        display: flex;
        margin: 0;
        justify-content: space-around;
    }

    .nav-menu li {
        border-bottom: none;
        flex: 1;
        text-align: center;
    }

    .nav-menu a {
        padding: 15px 10px;
    }

    .nav-menu a:hover {
        padding-left: 10px;
        background-color: #1619FF;
```

```
    }
}
```

🧪 6. Herramientas de Testing

DevTools del Navegador

```
// Simular diferentes dispositivos en DevTools
// Chrome/Firefox/Safari: F12 → Device Toolbar

// Dispositivos comunes para testing:
// - iPhone SE (375x667)
// - iPhone 12 Pro (390x844)
// - iPad (768x1024)
// - Desktop (1920x1080)

// Verificar viewport actual
console.log('Viewport:', window.innerWidth + 'x' + window.innerHeight);
console.log('Device Pixel Ratio:', window.devicePixelRatio);
```

🛠️ 1. Introducción a las Herramientas Modernas de Layout

¿Por qué Necesitamos Flexbox y Grid?

Problemas con Métodos Tradicionales

```
/* ❌ Problemas con floats */
.columna {
    float: left;
    width: 33.33%;
}
/* Problemas: clearfix, cálculos complejos, difícil centrado */

/* ❌ Problemas con posicionamiento */
.centrado {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
/* Problemas: sacar elementos del flujo, posicionamiento manual */
```

Soluciones Modernas

```
/* ✅ Con Flexbox: Centrado fácil */
.container {
    display: flex;
    justify-content: center;
    align-items: center;
}

/* ✅ Con Grid: Layouts complejos simples */
.layout {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: auto 1fr auto;
}
```

Comparación Rápida

Método	Mejor Para	Dificultad	Soporte
Floats	Texto alrededor de imágenes	⭐⭐⭐	100%
Flexbox	Componentes, barras de navegación	⭐⭐	98%+
Grid	Layouts de página completa	⭐⭐	95%+

🤸 2. Flexbox: Fundamentos

¿Qué es Flexbox?

Flexbox (Flexible Box Layout) es un método de layout que permite distribuir espacio y alinear elementos en un contenedor, incluso cuando su tamaño es desconocido o dinámico.

Conceptos Básicos

Contenedor y Elementos

```
<!-- HTML Structure -->
<div class="flex-container">
    <div class="flex-item">Elemento 1</div>
    <div class="flex-item">Elemento 2</div>
    <div class="flex-item">Elemento 3</div>
</div>
```

```
/* Activar Flexbox */
.flex-container {
    display: flex;  /* ¡Esto activa Flexbox! */
}

/* Los elementos hijos automáticamente se vuelven flex items */
.flex-item {
    background: #f0f0f0;
    padding: 1rem;
    margin: 0.5rem;
    border-radius: 4px;
}
```

Ejes Principales

> **Eje Principal (Main Axis):** La dirección principal en la que se colocan los elementos
> **Eje Cruzado (Cross Axis):** Siempre perpendicular al eje principal

Propiedades del Contenedor (Flex Container)

1. flex-direction

```
.container {
    display: flex;
    flex-direction: row;            /* → horizontal (por defecto) */
    /* flex-direction: row-reverse; /* ← horizontal inverso */
    /* flex-direction: column;      /* ↓ vertical */
    /* flex-direction: column-reverse; /* ↑ vertical inverso */
}
```

2. justify-content (Eje Principal)

```
.container {
    display: flex;
    justify-content: flex-start;    /* inicio (por defecto) */
    /* justify-content: flex-end;   /* final */
    /* justify-content: center;     /* centrado */
    /* justify-content: space-between; /* espacio entre elementos */
    /* justify-content: space-around;  /* espacio alrededor */
    /* justify-content: space-evenly;  /* espacio uniforme */
}
```

3. align-items (Eje Cruzado)

```
.container {
    display: flex;
    align-items: stretch;           /* estirar (por defecto) */
    /* align-items: flex-start;     /* inicio */
    /* align-items: flex-end;       /* final */
    /* align-items: center;         /* centrado */
    /* align-items: baseline;       /* línea base del texto */
}
```

4. flex-wrap

```
.container {
    display: flex;
    flex-wrap: nowrap;            /* no envolver (por defecto) */
    /* flex-wrap: wrap;           /* sí, hacia abajo */
    /* flex-wrap: wrap-reverse;   /* sí, hacia arriba */
}
```

Propiedades de los Elementos (Flex Items)

1. flex-grow

```
.item {
    /* ¿Cuánto puede crecer este elemento? */
    flex-grow: 0;       /* no crece (por defecto) */
    flex-grow: 1;       /* crece proporcionalmente */
    flex-grow: 2;       /* crece el doble que elementos con flex-grow: 1 */
}
```

2. flex-shrink

```
.item {
    /* ¿Cuánto puede encogerse este elemento? */
    flex-shrink: 1;     /* puede encogerse (por defecto) */
    flex-shrink: 0;     /* no se encoge nunca */
    flex-shrink: 2;     /* se encoge más que otros */
}
```

3. flex-basis

```
.item {
    /* Tamaño base antes de distribución de espacio */
    flex-basis: auto;       /* tamaño natural (por defecto) */
    flex-basis: 200px;      /* tamaño fijo en píxeles */
    flex-basis: 30%;        /* porcentaje del contenedor */
}
```

4. Shorthand: flex

```
.item {
    /* flex: grow shrink basis */
    flex: 1;            /* flex: 1 1 0% (flexible) */
    flex: auto;         /* flex: 1 1 auto (flexible) */
    flex: none;         /* flex: 0 0 auto (no flexible) */
    flex: 0 1 200px;    /* valores específicos */
}
```

✨ 3. Flexbox en la Práctica

Ejemplo 1: Navegación Horizontal

```
<nav class="navbar">
    <div class="brand">Mi Sitio</div>
    <ul class="nav-menu">
        <li><a href="#">Inicio</a></li>
        <li><a href="#">Productos</a></li>
        <li><a href="#">Contacto</a></li>
    </ul>
</nav>
```

```css
.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem;
    background: #343a40;
    color: white;
}

.nav-menu {
    display: flex;
    list-style: none;
    gap: 2rem;
    margin: 0;
    padding: 0;
}

.nav-menu a {
    color: white;
    text-decoration: none;
}

/* Resultado: Brand a la izquierda, menú a la derecha */
```

Ejemplo 2: Grid de Tarjetas

```html
<div class="card-container">
    <div class="card">Tarjeta 1</div>
    <div class="card">Tarjeta 2</div>
    <div class="card">Tarjeta 3</div>
    <div class="card">Tarjeta 4</div>
</div>
```

```css
.card-container {
    display: flex;
    flex-wrap: wrap;
    gap: 1rem;
    padding: 1rem;
}

.card {
    /* Cada tarjeta ocupa como mínimo 250px y crece */
    flex: 1 1 250px;
    background: #f8f9fa;
    padding: 1.5rem;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

/* En pantallas pequeñas, cada tarjeta ocupa toda la fila */
@media (max-width: 600px) {
    .card {
        flex-basis: 100%;
    }
}
```

Ejemplo 3: Centrado Perfecto

```html
<div class="container">
    <div class="centered-box">
        <h2>¡Perfectamente Centrado!</h2>
        <p>Vertical y horizontalmente</p>
    </div>
</div>
```

```css
.container {
    display: flex;
    justify-content: center;    /* centrado horizontal */
    align-items: center;        /* centrado vertical */
    min-height: 100vh;          /* altura mínima de pantalla */
}

.centered-box {
    background: white;
    padding: 2rem;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    text-align: center;
}
```

Ejemplo 4: Layout con Sidebar

```html
<div class="layout">
    <aside class="sidebar">Sidebar</aside>
    <main class="main-content">Contenido Principal</main>
</div>
```

```css
.layout {
    display: flex;
    min-height: 100vh;
}

.sidebar {
    flex: 0 0 250px;            /* ancho fijo de 250px */
    background: #34495e;
    color: white;
    padding: 1rem;
}

.main-content {
    flex: 1;                    /* ocupa todo el espacio restante */
    padding: 1rem;
}

/* Responsive: sidebar arriba en móviles */
@media (max-width: 768px) {
    .layout {
        flex-direction: column;
    }

    .sidebar {
        flex: 0 0 auto;
    }
}
```

▦ 4. CSS Grid: Fundamentos

¿Qué es CSS Grid?

CSS Grid es un sistema de layout bidimensional que te permite crear layouts complejos con control total sobre filas y columnas.

Conceptos Básicos

Contenedor y Elementos

```html
<!-- HTML Structure -->
<div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
</div>
```

```css
/* Activar Grid */
.grid-container {
    display: grid;  /* ¡Esto activa Grid! */
}
```

Definición de Columnas y Filas

1. grid-template-columns

```css
.grid-container {
    display: grid;

    /* Valores fijos */
    grid-template-columns: 200px 300px 100px;

    /* Valores flexibles con fr (fracciones) */
    grid-template-columns: 1fr 2fr 1fr;  /* proporción 1:2:1 */

    /* Combinación */
    grid-template-columns: 200px 1fr 100px;

    /* Repetición */
    grid-template-columns: repeat(3, 1fr);       /* 3 columnas iguales */
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
}
```

2. grid-template-rows

```css
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);

    /* Filas explícitas */
    grid-template-rows: 100px 200px auto;

    /* Filas automáticas para contenido extra */
    grid-auto-rows: 150px;  /* altura mínima para filas automáticas */
}
```

3. Espaciado (Gap)

```css
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);

    /* Espaciado entre elementos */
    gap: 20px;                      /* mismo espacio para filas y columnas */
    /* row-gap: 20px; column-gap: 10px; /* diferentes espacios */
}
```

Posicionamiento de Elementos

1. Por Líneas de Grid

```css
.grid-item {
    /* Posicionamiento por líneas (empiezan en 1) */
    grid-column-start: 1;
    grid-column-end: 3;       /* ocupa desde columna 1 hasta 3 */

    grid-row-start: 2;
    grid-row-end: 4;          /* ocupa desde fila 2 hasta 4 */

    /* Shorthand */
    grid-column: 1 / 3;       /* columnas 1 a 3 */
    grid-row: 2 / 4;          /* filas 2 a 4 */

    /* Con span (cantidad de espacios) */
    grid-column: 1 / span 2;  /* empieza en 1, ocupa 2 espacios */
}
```

2. Por Áreas Nombradas

```css
/* Definir áreas en el contenedor */
.grid-container {
    display: grid;
    grid-template-columns: 1fr 3fr 1fr;
    grid-template-rows: auto 1fr auto;
    grid-template-areas:
        "header header header"
        "sidebar main aside"
        "footer footer footer";
}

/* Asignar elementos a áreas */
.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.main { grid-area: main; }
.aside { grid-area: aside; }
.footer { grid-area: footer; }
```

Alineación en Grid

```css
.grid-container {
    display: grid;

    /* Alineación de todos los elementos */
    justify-items: center;    /* horizontal: start, end, center, stretch */
    align-items: center;      /* vertical: start, end, center, stretch */

    /* Alineación del grid completo dentro del contenedor */
    justify-content: center;  /* horizontal */
    align-content: center;    /* vertical */
}

/* Alineación individual de elementos */
.grid-item {
    justify-self: start;      /* horizontal para este elemento */
    align-self: end;          /* vertical para este elemento */
}
```

5. Grid en la Práctica

Ejemplo 1: Layout de Página Completa

```html
<div class="page-layout">
    <header class="header">Header</header>
    <nav class="navbar">Navigation</nav>
    <aside class="sidebar">Sidebar</aside>
    <main class="main">Main Content</main>
    <aside class="ads">Ads</aside>
    <footer class="footer">Footer</footer>
</div>
```

```css
.page-layout {
    display: grid;
    min-height: 100vh;
    grid-template-columns: 200px 1fr 150px;
    grid-template-rows: auto auto 1fr auto;
    grid-template-areas:
        "header header header"
        "navbar navbar navbar"
        "sidebar main ads"
        "footer footer footer";
    gap: 10px;
}

.header {
    grid-area: header;
    background: #2c3e50;
    color: white;
    padding: 1rem;
}

.navbar {
    grid-area: navbar;
    background: #34495e;
    color: white;
    padding: 0.5rem 1rem;
}

.sidebar {
    grid-area: sidebar;
    background: #ecf0f1;
    padding: 1rem;
}

.main {
    grid-area: main;
    background: white;
    padding: 1rem;
}

.ads {
    grid-area: ads;
    background: #f8f9fa;
    padding: 1rem;
}

.footer {
    grid-area: footer;
    background: #2c3e50;
    color: white;
    padding: 1rem;
}
```

Ejemplo 2: Galería de Imágenes Responsive

```html
<div class="image-gallery">
    <img src="image1.jpg" alt="Imagen 1">
    <img src="image2.jpg" alt="Imagen 2">
    <img src="image3.jpg" alt="Imagen 3">
    <img src="image4.jpg" alt="Imagen 4">
    <img src="image5.jpg" alt="Imagen 5">
    <img src="image6.jpg" alt="Imagen 6">
</div>
```

```css
.image-gallery {
    display: grid;
    /* Columnas automáticas, mínimo 250px, máximo 1fr */
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    padding: 20px;
}

.image-gallery img {
    width: 100%;
    height: 200px;
    object-fit: cover;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
    transition: transform 0.3s ease;
}

.image-gallery img:hover {
    transform: scale(1.05);
}

/* Destacar primera imagen */
.image-gallery img:first-child {
    grid-column: span 2;  /* ocupa 2 columnas */
    height: 300px;
}
```

Ejemplo 3: Cards con Grid Complejo

```html
<div class="cards-grid">
    <article class="card featured">
        <h3>Card Destacado</h3>
        <p>Contenido especial...</p>
    </article>
    <article class="card">
        <h3>Card Normal</h3>
        <p>Contenido...</p>
    </article>
    <article class="card">
        <h3>Card Normal</h3>
        <p>Contenido...</p>
    </article>
    <article class="card wide">
        <h3>Card Ancho</h3>
        <p>Contenido más amplio...</p>
    </article>
</div>
```

```css
.cards-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 20px;
    padding: 20px;
}

.card {
    background: white;
    padding: 1.5rem;
    border-radius: 10px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    border: 1px solid #e1e8ed;
}

.card h3 {
    margin-top: 0;
    color: #2c3e50;
}

/* Card destacado: más alto */
.card.featured {
    grid-row: span 2;
    background: linear-gradient(135deg, #1619FF 0%, #FA01FA 100%);
    color: white;
}

/* Card ancho: ocupa 2 columnas si hay espacio */
.card.wide {
    grid-column: span 2;
}

/* Responsive: en pantallas pequeñas, todos los cards ocupan el ancho completo */
@media (max-width: 768px) {
    .cards-grid {
        grid-template-columns: 1fr;
    }

    .card.wide,
    .card.featured {
        grid-column: span 1;
        grid-row: span 1;
    }
}
```

Ejemplo 4: Layout Dashboard

```html
<div class="dashboard">
    <header class="dash-header">Dashboard</header>
    <div class="stat-card users">Usuarios</div>
    <div class="stat-card sales">Ventas</div>
    <div class="stat-card revenue">Ingresos</div>
    <div class="chart main-chart">Gráfico Principal</div>
    <div class="chart pie-chart">Gráfico Circular</div>
    <div class="recent-activity">Actividad Reciente</div>
</div>
```

```css
.dashboard {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    grid-template-rows: auto auto 1fr 1fr;
    gap: 20px;
    padding: 20px;
    min-height: 100vh;
}

.dash-header {
    grid-column: 1 / -1;  /* ocupa todas las columnas */
    background: #2c3e50;
    color: white;
    padding: 1rem;
    border-radius: 8px;
}

.stat-card {
    background: white;
    padding: 1.5rem;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    text-align: center;
}

.main-chart {
    grid-column: 1 / 4;  /* ocupa 3 columnas */
    grid-row: 3 / 5;     /* ocupa 2 filas */
    background: white;
    border-radius: 8px;
    padding: 1rem;
}

.pie-chart {
    grid-column: 4;
    grid-row: 3;
    background: white;
    border-radius: 8px;
    padding: 1rem;
}

.recent-activity {
    grid-column: 4;
    grid-row: 4;
    background: white;
    border-radius: 8px;
    padding: 1rem;
}
```

🆚 6. Flexbox vs Grid: ¿Cuándo Usar Cada Uno?

Conceptos Clave

> **Flexbox:** Diseño unidimensional (una dirección: fila O columna)
> **Grid:** Diseño bidimensional (dos direcciones: filas Y columnas)

Cuándo Usar Flexbox

✅ Casos Ideales para Flexbox

- **Navegación horizontal:** Menús, barras de herramientas
- **Centrado de elementos:** Modal, contenido en el viewport
- **Distribución de espacio:** Botones, elementos con espaciado automático
- **Componentes pequeños:** Cards, formularios, botones
- **Layouts simples:** Header con logo y menú, sidebar + contenido
- **Contenido dinámico:** Cuando no sabes cuántos elementos tendrás

Ejemplo: Barra de Navegación con Flexbox

```
/* Perfecto para Flexbox */
.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.nav-menu {
    display: flex;
    gap: 1rem;
}

/* Resultado: Logo a la izquierda, menú a la derecha, todo centrado verticalmente */
```

Cuándo Usar Grid

✅ Casos Ideales para Grid

- **Layouts completos de página:** Header, sidebar, main, footer
- **Galerías de imágenes:** Grids de fotos, portfolios
- **Dashboards:** Múltiples widgets con diferentes tamaños
- **Formularios complejos:** Con etiquetas y campos alineados
- **Magazines layouts:** Diferentes secciones con tamaños específicos
- **Cuando necesitas control exacto:** Posicionamiento preciso

Ejemplo: Layout de Página con Grid

```
/* Perfecto para Grid */
.page-layout {
    display: grid;
    grid-template-areas:
        "header header header"
        "sidebar main aside"
        "footer footer footer";
    grid-template-columns: 200px 1fr 200px;
    grid-template-rows: auto 1fr auto;
}

/* Resultado: Layout completo y estructurado */
```

Comparación Directa

Aspecto	Flexbox	Grid
Dimensiones	1D (fila O columna)	2D (filas Y columnas)
Mejor para	Componentes, distribución	Layouts, estructuras
Contenido	Dinámico, desconocido	Conocido, estructurado
Control	Content-first (desde el contenido)	Layout-first (desde el diseño)
Aprendizaje	Más simple	Más complejo

¿Se Pueden Combinar?

> **¡Absolutamente sí!** Es común y recomendado usar ambos en el mismo proyecto.

Ejemplo: Combinando Grid y Flexbox

```css
/* Grid para el layout general */
.page {
    display: grid;
    grid-template-areas:
        "header"
        "main"
        "footer";
    grid-template-rows: auto 1fr auto;
}

/* Flexbox para la navegación dentro del header */
.header {
    grid-area: header;
    display: flex;              /* ¡Flexbox dentro de Grid! */
    justify-content: space-between;
    align-items: center;
}

/* Flexbox para las tarjetas dentro del main */
.cards-container {
    display: flex;              /* ¡Otro Flexbox! */
    flex-wrap: wrap;
    gap: 1rem;
}
```

Guía de Decisión Rápida

Pregúntate:

1. **¿Es un layout completo de página?** → **Grid**
2. **¿Es una barra de navegación?** → **Flexbox**
3. **¿Necesito posicionar elementos en filas Y columnas?** → **Grid**
4. **¿Solo necesito alinear elementos en una dirección?** → **Flexbox**
5. **¿No sé cuántos elementos tendré?** → **Flexbox**
6. **¿Necesito control exacto del layout?** → **Grid**

📱 7. Responsive Design con Flexbox y Grid

Responsive Flexbox

Patrón 1: Navegación Responsive

```html
<nav class="responsive-nav">
    <div class="brand">Mi Marca</div>
    <ul class="nav-menu">
        <li><a href="#">Inicio</a></li>
        <li><a href="#">Productos</a></li>
        <li><a href="#">Contacto</a></li>
    </ul>
</nav>
```

```
/* Desktop: horizontal */
.responsive-nav {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem;
    background: #2c3e50;
    color: white;
}

.nav-menu {
    display: flex;
    list-style: none;
    gap: 2rem;
    margin: 0;
    padding: 0;
}

/* Tablet y móvil: vertical */
@media (max-width: 768px) {
    .responsive-nav {
        flex-direction: column;
        gap: 1rem;
    }

    .nav-menu {
        flex-direction: column;
        text-align: center;
        gap: 1rem;
    }
}
```

Patrón 2: Tarjetas Flexibles

```
.cards-flex {
    display: flex;
    flex-wrap: wrap;
    gap: 1rem;
    padding: 1rem;
}

.card {
    /* Mínimo 280px, máximo crece proporcionalmente */
    flex: 1 1 280px;
    background: white;
    padding: 1.5rem;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

/* Móvil: una tarjeta por fila */
@media (max-width: 600px) {
    .card {
        flex-basis: 100%;
    }
}
```

Patrón 3: Hero Section Centrado

```css
.hero {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    padding: 2rem;
    text-align: center;
    background: linear-gradient(135deg, #1619FF 0%, #FA01FA 100%);
    color: white;
}

.hero h1 {
    font-size: clamp(2rem, 5vw, 4rem);  /* Tamaño fluido */
    margin-bottom: 1rem;
}

.hero p {
    font-size: clamp(1rem, 2.5vw, 1.5rem);
    max-width: 600px;
}
```

Responsive Grid

Patrón 1: Layout de Página Adaptativo

```css
.responsive-layout {
    display: grid;
    min-height: 100vh;
    gap: 1rem;
}

/* Desktop: layout completo */
@media (min-width: 1024px) {
    .responsive-layout {
        grid-template-columns: 250px 1fr 200px;
        grid-template-rows: auto 1fr auto;
        grid-template-areas:
            "header header header"
            "sidebar main aside"
            "footer footer footer";
    }
}

/* Tablet: sin aside */
@media (min-width: 768px) and (max-width: 1023px) {
    .responsive-layout {
        grid-template-columns: 200px 1fr;
        grid-template-rows: auto 1fr auto;
        grid-template-areas:
            "header header"
            "sidebar main"
            "footer footer";
    }
}

/* Móvil: layout vertical */
@media (max-width: 767px) {
    .responsive-layout {
        grid-template-columns: 1fr;
        grid-template-areas:
            "header"
            "main"
            "sidebar"
            "footer";
    }
}
```

Patrón 2: Auto-fit Gallery

```css
.auto-gallery {
    display: grid;
    /* ¡La magia está aquí! */
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 1rem;
    padding: 1rem;
}

.gallery-item {
    background: white;
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
}

.gallery-item img {
    width: 100%;
    height: 200px;
    object-fit: cover;
}

/* No necesita media queries! Se adapta automáticamente */
```

Patrón 3: Dashboard Responsive

```css
.responsive-dashboard {
    display: grid;
    gap: 1rem;
    padding: 1rem;
}

/* Desktop: dashboard completo */
@media (min-width: 1024px) {
    .responsive-dashboard {
        grid-template-columns: repeat(4, 1fr);
        grid-template-areas:
            "header header header header"
            "stats1 stats2 stats3 stats4"
            "chart chart chart sidebar"
            "chart chart chart sidebar";
    }
}

/* Tablet: layout compacto */
@media (min-width: 768px) and (max-width: 1023px) {
    .responsive-dashboard {
        grid-template-columns: repeat(3, 1fr);
        grid-template-areas:
            "header header header"
            "stats1 stats2 stats3"
            "chart chart sidebar"
            "chart chart sidebar";
    }
}

/* Móvil: layout lineal */
@media (max-width: 767px) {
    .responsive-dashboard {
        grid-template-columns: 1fr;
        grid-template-areas:
            "header"
            "stats1"
            "stats2"
            "stats3"
            "chart"
            "sidebar";
    }
}
```

Técnicas Avanzadas

1. Container Queries (Futuro)

```
/* Aún experimental, pero viene pronto */
.card-container {
    container-type: inline-size;
}

@container (min-width: 400px) {
    .card {
        display: grid;
        grid-template-columns: 1fr 2fr;
    }
}
```

2. Subgrid (Próximamente)

```
/* Permitirá que grids anidados hereden las líneas del padre */
.parent-grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
}

.child-grid {
    display: grid;
    grid-template-columns: subgrid;  /* Hereda las columnas */
}
```

3. Aspect Ratio para Layouts

```
.video-container {
    aspect-ratio: 16 / 9;  /* Siempre mantiene proporción */
    width: 100%;
}

.card-image {
    aspect-ratio: 1;       /* Siempre cuadrado */
    object-fit: cover;
}
```

👍 8. Mejores Prácticas

Prácticas para Flexbox

✅ Recomendaciones

- **Usa flex shorthand:** `flex: 1` en lugar de propiedades individuales
- **Piensa en el eje principal:** ¿Es horizontal o vertical?
- **Usa gap:** Mejor que margin para espaciado
- **Combina con media queries:** Para cambiar direcciones

```css
/* ✅ Bueno: Usa flex shorthand */
.item {
    flex: 1;  /* flex: 1 1 0% */
}

/* ❌ Evitar: Propiedades separadas innecesariamente */
.item {
    flex-grow: 1;
    flex-shrink: 1;
    flex-basis: 0%;
}

/* ✅ Bueno: Usa gap */
.container {
    display: flex;
    gap: 1rem;
}

/* ❌ Evitar: Margin en cada elemento */
.item {
    margin-right: 1rem;
}
```

❌ Errores Comunes

- **Confundir ejes:** justify-content vs align-items
- **No entender flex-basis:** Es el tamaño inicial, no final
- **Usar flexbox para layouts 2D:** Grid es mejor

Prácticas para Grid

✅ Recomendaciones

- **Define áreas semánticamente:** header, main, footer
- **Usa repeat() y auto-fit:** Para layouts flexibles
- **Planifica tu grid:** Dibuja primero, codifica después
- **Usa minmax():** Para tamaños responsivos

```css
/* ✅ Bueno: Áreas semánticas */
.layout {
    grid-template-areas:
        "header header"
        "sidebar main"
        "footer footer";
}

/* ✅ Bueno: Responsive automático */
.gallery {
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
}

/* ✅ Bueno: Tamaños flexibles */
.container {
    grid-template-columns: minmax(200px, 1fr) 3fr minmax(150px, 1fr);
}
```

❌ Errores Comunes

- **Usar Grid para todo:** A veces Flexbox es más simple
- **Olvidar definir filas:** Pueden crecer automáticamente
- **Posicionamiento manual excesivo:** Aprovecha el flujo automático

Rendimiento y Optimización

Optimización de CSS

```
/* ✅ Bueno: Especificidad baja */
.card { display: flex; }

/* ❌ Evitar: Especificidad alta */
.page .section .container .card { display: flex; }

/* ✅ Bueno: Agrupa propiedades relacionadas */
.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 1rem;
}

/* ❌ Evitar: Propiedades separadas */
.container { display: grid; }
.container { grid-template-columns: repeat(3, 1fr); }
.container { gap: 1rem; }
```

Optimización de Rendimiento

- **Evita layouts complejos anidados:** Pueden ser costosos
- **Usa will-change con cuidado:** Solo para elementos animados
- **Mide el rendimiento:** DevTools → Performance

Herramientas de Desarrollo

DevTools para Flexbox/Grid

Chrome/Edge DevTools:
• Inspector de Grid: muestra líneas y áreas
• Inspector de Flexbox: visualiza dirección y wrap
• Layout panel: para debugging avanzado

Firefox DevTools:
• Grid Inspector: el más avanzado
• Flexbox Inspector: excelente para debugging
• Layout tab: herramientas dedicadas

Herramientas Online

- **Flexbox Froggy:** Juego para aprender Flexbox
- **Grid Garden:** Juego para aprender Grid
- **CSS Grid Generator:** Generador visual de Grid
- **Flexbox Playground:** Experimentar con propiedades

Checklist de Implementación

Antes de Implementar:

- ☐ ¿Necesito layout 1D o 2D?
- ☐ ¿El contenido es dinámico o conocido?
- ☐ ¿Es un componente o un layout?
- ☐ ¿Necesito soporte en navegadores antiguos?

Durante la Implementación:

- ☐ Usar nombres semánticos para áreas
- ☐ Implementar mobile-first
- ☐ Probar en diferentes tamaños de pantalla
- ☐ Verificar en múltiples navegadores

Después de Implementar:

- ☐ Validar HTML y CSS
- ☐ Probar accesibilidad
- ☐ Medir rendimiento
- ☐ Documentar decisiones de diseño

🌐 9. Compatibilidad con Navegadores

Soporte Actual (2024)

Navegador	Flexbox	Grid	Notas
Chrome 29+	✅	✅	Soporte completo
Firefox 28+	✅	✅	Excelente DevTools
Safari 9+	✅	⚠️	Grid desde v10.1
Edge 12+	✅	⚠️	Grid desde v16
IE 10-11	⚠️	❌	Flexbox con prefijos

Estrategias de Fallback

1. Detección de Soporte

```
/* CSS Feature Queries */
@supports (display: grid) {
    .container {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
    }
}

/* Si no soporta Grid, usa Flexbox */
@supports not (display: grid) {
    .container {
        display: flex;
        flex-wrap: wrap;
    }

    .item {
        flex: 1 1 300px;
    }
}
```

2. Progressive Enhancement

```css
/* Enfoque: desde lo básico hasta lo moderno */

/* 1. Base: funciona en todos los navegadores */
.container {
    overflow: hidden;  /* clearfix */
}

.item {
    float: left;
    width: 33.333%;
    padding: 1rem;
    box-sizing: border-box;
}

/* 2. Mejora: si soporta Flexbox */
@supports (display: flex) {
    .container {
        display: flex;
        flex-wrap: wrap;
    }

    .item {
        float: none;    /* anula float */
        flex: 1 1 300px;
    }
}

/* 3. Mejora final: si soporta Grid */
@supports (display: grid) {
    .container {
        display: grid;
        grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
        gap: 1rem;
    }

    .item {
        padding: 0;     /* anula padding, usa gap */
    }
}
```

3. Prefijos para Flexbox Antiguo

```css
/* Para IE 10-11 y versiones muy antiguas */
.container {
    display: -webkit-box;      /* 2009 - muy antiguo */
    display: -webkit-flex;     /* 2012 - con prefijos */
    display: -ms-flexbox;      /* IE 10-11 */
    display: flex;             /* estándar moderno */

    -webkit-box-orient: horizontal;
    -webkit-box-direction: normal;
    -webkit-flex-direction: row;
    -ms-flex-direction: row;
    flex-direction: row;
}

/* Usa Autoprefixer para generar esto automáticamente */
```

Problemas Conocidos y Soluciones

Internet Explorer 11

> ### Problemas con Flexbox en IE11:
>
> - flex-basis no funciona correctamente
> - min-height/min-width pueden causar problemas
> - flex-wrap: wrap tiene bugs

```
/* Soluciones para IE11 */

/* Problema: flex-basis no funciona */
/* ❌ No funciona en IE11 */
.item {
    flex: 1 1 300px;
}

/* ✅ Funciona en IE11 */
.item {
    flex: 1 1 auto;
    width: 300px;        /* usa width/height en lugar de flex-basis */
}

/* Problema: min-height en flex containers */
/* ❌ No funciona en IE11 */
.container {
    display: flex;
    min-height: 100vh;
}

/* ✅ Funciona en IE11 */
.container {
    display: flex;
    height: 100vh;       /* usa height fija */
}
```

Safari

> ### Problemas con Grid en Safari:
>
> - gap no soportado en versiones antiguas
> - Algunos problemas con auto-fit/auto-fill

```
/* Fallback para gap en Safari antiguo */
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 1rem;                /* moderno */
    grid-gap: 1rem;             /* fallback para Safari */
}
```

Herramientas para Compatibilidad

1. Autoprefixer

```
/* Tu escribes: */
.container {
    display: flex;
    flex-direction: column;
}

/* Autoprefixer genera: */
.container {
    display: -webkit-box;
    display: -ms-flexbox;
    display: flex;
    -webkit-box-orient: vertical;
    -webkit-box-direction: normal;
    -ms-flex-direction: column;
    flex-direction: column;
}
```

2. PostCSS Grid Fallback

```
/* Plugin para generar fallbacks automáticos de Grid */
npm install postcss-grid-kiss

/* Tu escribes Grid moderno, el plugin genera fallbacks */
```

3. Verificación de Soporte

- **Can I Use:** caniuse.com - Tablas de compatibilidad
- **MDN Browser Compatibility:** Datos detallados
- **Browserstack:** Pruebas en navegadores reales

Estrategia Recomendada 2024

> ### Enfoque Pragmático:
>
> 1. **Usa Flexbox y Grid libremente** - Soporte excelente en navegadores modernos
> 2. **Ignora IE11** - A menos que sea requisito específico del cliente
> 3. **Proporciona fallbacks básicos** - Con @supports cuando sea necesario
> 4. **Usa Autoprefixer** - Para manejar prefijos automáticamente
> 5. **Prueba en navegadores reales** - No solo en DevTools

➡️ 10. Resumen y Próximos Pasos

Lo que Hemos Aprendido

✅ Conceptos Clave Dominados:

- **Flexbox:** Layout unidimensional para componentes y alineación
- **CSS Grid:** Layout bidimensional para estructuras complejas
- **Cuándo usar cada uno:** Criterios de decisión claros
- **Patrones responsive:** Técnicas modernas y eficientes
- **Mejores prácticas:** Código limpio y mantenible
- **Compatibilidad:** Estrategias de fallback y soporte

Evolución del Layout CSS

Era	Tecnología	Estado
1990s-2000s	Tables, Float	📱 Obsoleto
2010s	Flexbox	✅ **Estándar**
2017+	CSS Grid	✅ **Estándar**
2023+	Container Queries	🚀 Emergente