DISEÑO Y UBICACIÓN DE CONTENIDOS

LA IMPORTANCIA DEL DISEÑO Y UBICACIÓN DEL CONTENIDO

- **El diseño afecta directamente a la experiencia de usuario.**
- **Un contenido mal ubicado puede generar confusión y abandono del sitio.**
- **Los usuarios toman decisiones rápidas basadas en la apariencia y disposición del contenido.**
- **Un diseño ordenado guía la atención hacia las áreas importantes.**
- El diseño influye en la **tasa de rebote** y **conversiones**

ASPECTOS CLAVE A CONSIDERAR

- **Jerarquía Visual**: Organizar los contenidos de manera que el usuario pueda comprender la importancia de cada sección de un vistazo.
- **Patrones de Diseño**: Los usuarios suelen seguir patrones de escaneo al navegar por una página.
- **Espaciado y Consistencia**: El uso correcto del espacio en blanco ayuda a que la página sea más fácil de navegar.

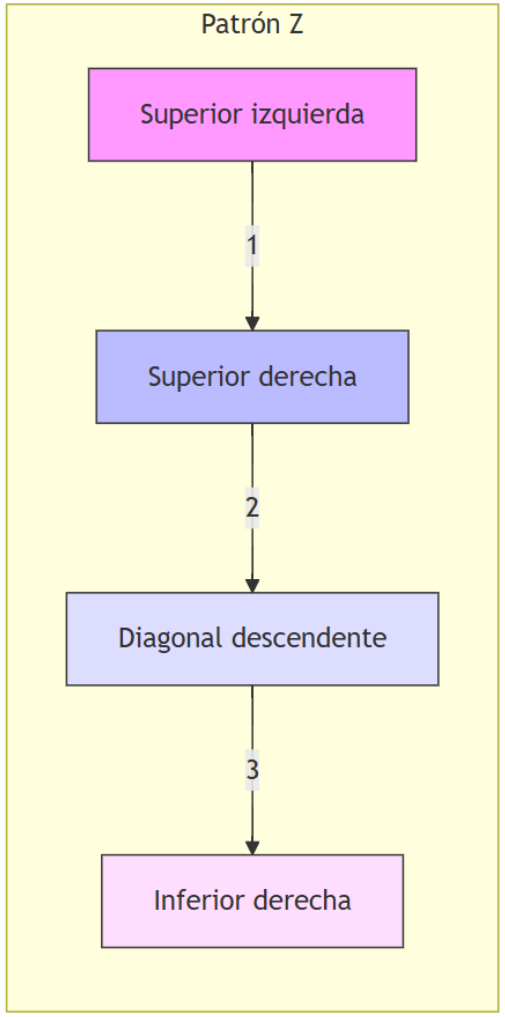
JERARQUÍA VISUAL: ORGANIZACIÓN DEL CONTENIDO

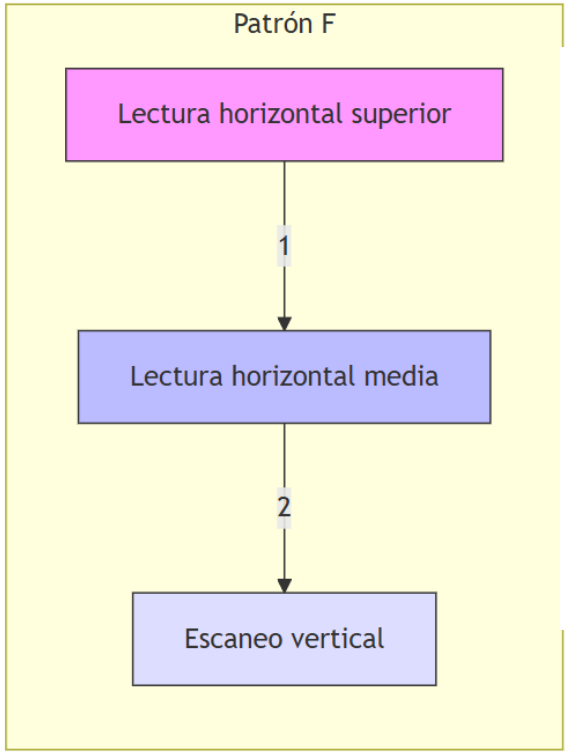
- **Destacar elementos clave con tamaños, colores y alineaciones.**
- **Usar títulos grandes para encabezados y fuentes más pequeñas para el texto secundario.**
- **Aplicar colores contrastantes para resaltar secciones importantes.**
- **Distribuir el contenido usando márgenes y espacios en blanco.**

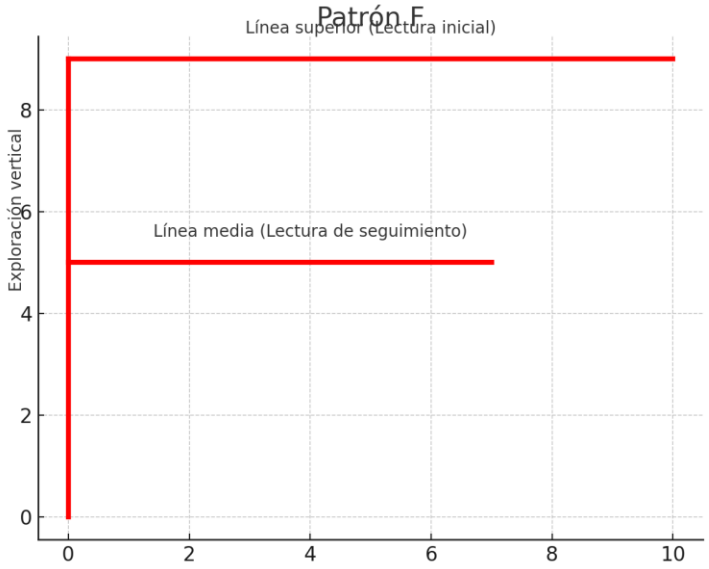
PATRONES DE DISEÑO: PATRÓN F Y PATRÓN Z

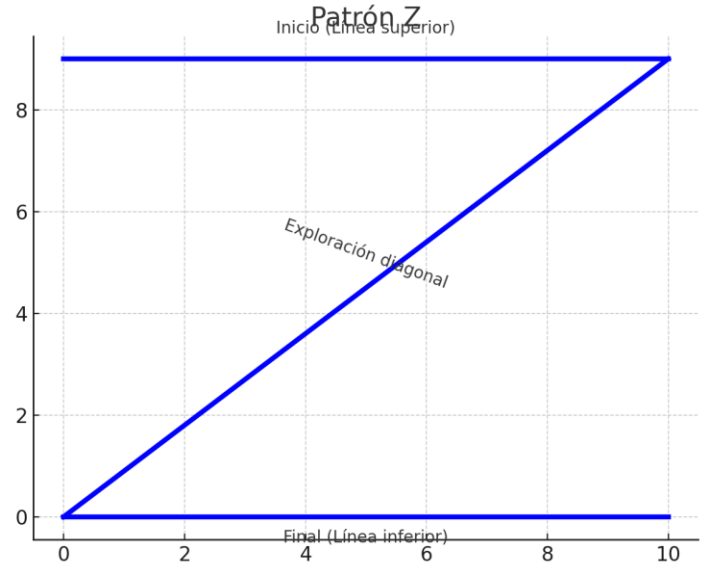
- **Patrón F:**
  - Escaneo en forma de “F”.
  - Común en **blogs, noticias y sitios con mucho texto**.
  - Utilizado principalmente en páginas con mucho contenido textual.
- **Patrón Z:**
  - Escaneo en forma de “Z”.
  - Usado en **landing pages o páginas con llamados a la acción**.
  - Observado principalmente en páginas de destino o sitios donde se espera una acción del usuario.
- **Adaptar el contenido al patrón de lectura del usuario mejora la navegación.**

PATRONES DE DISEÑO: PATRÓN F Y PATRÓN Z









- Los usuarios **escanean las páginas en lugar de leerlas completamente**.
- El uso del **patrón Z** en **páginas de inicio o promoción de productos es recomendable**.

ESPACIADO Y CONSISTENCIA VISUAL

- **Utilizar márgenes y rellenos uniformes para separar elementos.**
- **Aplicar la misma tipografía y paleta de colores en todo el sitio.**
- **Evitar diseños recargados y dejar suficiente espacio en blanco.**
- **Un diseño consistente facilita la comprensión visual.**

(ver ejemplo **Espaciado y Consistencia Visual**)

HERRAMIENTAS DE CSS PARA EL DISEÑO DE CONTENIDOS

- Flexbox: Organiza elementos dentro de un contenedor y los alinea de manera eficiente.

```
.contenedor {
    display: flex;
    justify-content: space-between;
    align-items: center;
}
```

- Grid Layout: Crea diseños complejos con varias columnas y filas.

```
.contenedor {
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 20px;
}
```

- Box Model: Define el espacio alrededor de los elementos.

FLEXBOX: DISTRIBUCIÓN EFICIENTE DE ELEMENTOS

```
.contenedor {
    display: flex;
    justify-content: space-between;
    align-items: center;
}
```

- **Flexbox organiza los elementos horizontal o verticalmente.**
- **Alinea y distribuye espacio entre elementos automáticamente.**
- **Ideal para secciones de navegación y estructuras flexibles.**

(ver ejemplo Flexbox: Distribución Eficiente de Elementos)

CSS GRID LAYOUT: DISEÑO EN COLUMNAS Y FILAS

```
.contenedor {
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 20px;
}
```

- **Grid Layout organiza el contenido en cuadrículas.**
- **Permite crear diseños con múltiples columnas y filas.**
- **Aporta flexibilidad para diseños complejos y responsivos.**

DISEÑO RESPONSIVO CON MEDIA QUERIES

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

- **Media Queries adaptan el diseño a diferentes pantallas.**
- **Permite cambiar el diseño en función del tamaño del dispositivo.**
- **Clave para garantizar que el sitio se vea bien en móviles y tablets.**
- **Unidades Relativas**: Aseguran que los elementos se ajusten automáticamente al tamaño del contenedor o de la pantalla del usuario.
- (ver ejemplo Las **media queries** y las **unidades relativas**)

BUENAS PRÁCTICAS PARA DISEÑO Y UBICACIÓN DE CONTENIDOS

- **Usar consistencia visual en colores y tipografías.**
- **Agrupar elementos relacionados visualmente.**
- **Mantener espacio en blanco para evitar sobrecarga.**
- **Asegurarse de que los botones y enlaces sean claros y accesibles.**

OPTIMIZACIÓN DE CONTENIDOS

IMPORTANCIA DE LA OPTIMIZACIÓN DE CONTENIDOS

- **Carga Lenta = Pérdida de Usuarios.**
- **Un sitio rápido mejora la tasa de retención y conversión.**
- **El SEO se beneficia de páginas rápidas y bien estructuradas.**
- **Optimización = Menor consumo de recursos del servidor y navegador.**
- **(ver ejemplo** ejemplo de comparación de velocidad entre un sitio optimizado y uno que no lo está**)**

El 53% de los usuarios abandonan un sitio si tarda más de 3 segundos en cargar.
La optimización no es solo técnica, sino que también afecta **la percepción de calidad del usuario**.

BENEFICIOS CLAVE DE LA OPTIMIZACIÓN WEB

- **Mejora de la Velocidad de Carga**: Una página optimizada carga más rápidamente.
- **Mejora del SEO**: Los motores de búsqueda priorizan los sitios bien optimizados.
- **Reducción del Consumo de Recursos**: Optimizar los archivos ayuda a reducir el uso de recursos.
- **Mejora de la Experiencia de Usuario**: La optimización ayuda a crear una experiencia de usuario más fluida.

Un sitio optimizado tiene menos tasa de rebote y **más interacción del usuario**.

Google prioriza sitios **rápidos y bien estructurados**.

TÉCNICAS DE OPTIMIZACIÓN DE CONTENIDOS

- **Optimización de Imágenes**: Las imágenes suelen ser los elementos más pesados de una página web.
- **Minificación de Archivos CSS y JavaScript**: La minificación elimina caracteres innecesarios para reducir el tamaño de los archivos.
- **Caché del Navegador**: Configurar la caché del navegador permite almacenar recursos localmente.
- **Carga Asincrónica de JavaScript**: Utilizar atributos como **async** o **defer** para cargar scripts en segundo plano.
- **Optimización de Fuentes**: Utilizar solo las variantes de fuente necesarias y aplicar formatos modernos como WOFF2

OPTIMIZACIÓN DE IMÁGENES PARA LA WEB

- **Compresión sin pérdida (TinyPNG, Squoosh).**
- **Elegir formatos adecuados (JPEG, PNG, WebP).**
- **Definir dimensiones exactas según el contenedor.**

```
img {
    max-width: 100%;
    height: auto;
}
```

Las imágenes **son responsables de gran parte del peso de una página**.

La **compresión puede reducir un 70% el tamaño** de una imagen sin perder calidad.

Recomendar el uso del formato **WebP** para **ahorrar espacio**.

- Compresión de Imágenes: Utilizar herramientas como **TinyPNG** o **Squoosh**.
- Formatos Adecuados: Utilizar JPEG para fotografías, PNG para transparencias y WebP para buena calidad con tamaño reducido.
- Dimensiones Correctas: Asegurarse de que las imágenes no sean más grandes de lo necesario.

MINIFICACIÓN DE ARCHIVOS CSS Y JAVASCRIPT

- **Eliminar espacios en blanco y comentarios.**
- **Herramientas:**
  - **CSSNano (CSS)**
  - **UglifyJS (JavaScript)**

```
/* Sin minificar */
body {
    color: #333;
    background-color: white;
}

/* Minificado */
body{color:#333;background-color:#fff}
```

- Ejemplo sin minificar:

```
body {
    background-color: #ffffff;
    color: #333333;
}
```

- Ejemplo minificado:

```
body{background-color:#fff;color:#333}
```

CACHÉ DEL NAVEGADOR

- Configurar la caché del navegador permite almacenar recursos localmente.
- Ejemplo de configuración de caché en el servidor:

```
<meta http-equiv="Cache-Control" content="max-age=3600">
```

  - Reduce tiempos de carga en visitas repetidas.

Visualización de la Caché en el Navegador (Chrome):

1. **Abrir DevTools (F12 o Ctrl+Shift+I)**
2. Ir a la pestaña **Application** > **Cache** > **Cache Storage**.
3. Ahí se pueden ver los archivos almacenados y su duración.

CARGA ASINCRÓNICA DE JAVASCRIPT

- Utilizar atributos como **async** o **defer** para cargar scripts en segundo plano.
- Ejemplo:

```
<script src="script.js" defer></script>
```

Diferencias entre async y defer:

async:

- El script se descarga en paralelo al parseo del HTML.
- La ejecución del script se realiza tan pronto como termina de descargarse, interrumpiendo el parseo del HTML.
- Ideal para scripts independientes que no dependen de otros.

defer:

- El script se descarga en paralelo, pero se ejecuta después de que el HTML haya sido completamente parseado.
- Mantiene el orden de ejecución de los scripts deferidos.
- Perfecto para scripts que dependen del DOM completo.

CARGA ASINCRÓNICA DE JAVASCRIPT

Cuando el navegador "parsea" un archivo HTML:

1. **Lee el código fuente del documento HTML línea por línea.**
2. **Interpreta las etiquetas, atributos y contenido:**
   - Identifica qué elementos son encabezados, párrafos, imágenes, scripts, etc.
3. **Construye una estructura jerárquica (el DOM):**
   - Por ejemplo, una etiqueta que contiene otras etiquetas se convierte en un nodo principal con nodos secundarios.

El atributo async en una etiqueta <script> afecta la forma en que los scripts interactúan con el proceso de parseo:

- Normalmente, si el navegador encuentra una etiqueta <script> sin async ni defer, detiene el parseo del HTML para descargar y ejecutar el script, bloqueando temporalmente el resto de la página.
- Con async, el script se descarga en paralelo mientras el navegador sigue parseando el HTML. Una vez que el script termina de descargarse, se ejecuta inmediatamente, aunque el HTML aún no haya sido completamente procesado.

CARGA ASINCRÓNICA DE JAVASCRIPT

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Ejemplo de parseo</title>
    <script src="script.js" async></script>
</head>
<body>
    <h1>Hola mundo</h1>
    <p>Este texto se muestra después del parseo.</p>
</body>
</html>
```

- Encuentra <script src="script.js" async> y comienza a descargar el script en paralelo, sin detener el proceso de parseo.
- Mientras el script se descarga, el navegador sigue construyendo el DOM (el contenido del <body>).
- Cuando el script termina de descargarse, se ejecuta inmediatamente, sin esperar a que el resto del HTML sea completamente parseado

OPTIMIZACIÓN DE FUENTES

- Utilizar solo las variantes de fuente necesarias.
- Aplicar formatos modernos como WOFF2.
- Implementar carga condicional de fuentes y Fallbacks.

```
body {
    font-family: 'Roboto', Arial, sans-serif;
}
```

- (ver ejemplo **Optimización de Fuentes**)

OPTIMIZACIÓN SEO PARA MEJORAR EL RANKING

- **Uso de Etiquetas Meta y Títulos Descriptivos**: Proporcionar información a los motores de búsqueda.

  <meta name="description" content="Optimización web para mejorar SEO y velocidad de carga.">

- **Optimización de Contenidos con Palabras Clave**: Utilizar palabras clave relevantes.

- **Uso de Etiquetas Alt en Imágenes**: Ayudar a los motores de búsqueda y mejorar la accesibilidad.

  <img src="imagen.jpg" alt="Descripción de la imagen">

- **Estructuración del Contenido con Encabezados (H1, H2, H3)**: Ayudar a los motores de búsqueda y a los usuarios a comprender la estructura del contenido

  (ver ejemplo técnicas de optimización SEO)

MEJORA DE LA EXPERIENCIA DE USUARIO (UX)

- **Carga diferida (Lazy Loading):**

  <img src="imagen.jpg" loading="lazy" alt="Descripción de la imagen">

- **Optimización de formularios:**
  - Formularios simples y con validación.
- **Botones y enlaces accesibles:**

  button:hover {

  background-color: #007acc;

  }

  (ver ejemplo de técnica de mejora de la experiencia de usuario)