# 1. Introducción al Diseño Web Adaptativo

El diseño web adaptativo se centra en que las páginas web se ajusten automáticamente a las características de diferentes dispositivos, mejorando la experiencia de usuario en móviles, tabletas, computadoras y televisores inteligentes. Originalmente, algunas empresas creaban versiones separadas de un sitio para cada dispositivo, lo cual era costoso y difícil de mantener. Ahora, el enfoque moderno utiliza técnicas avanzadas como CSS3 media queries, cuadrículas flexibles, y contenido adaptativo para adaptar un solo diseño a diferentes resoluciones y tamaños de pantalla.

# 2. Estrategias de Diseño

### 2.1 Tipos de Diseño Web

- Diseño de ancho fijo: Tiene un control visual preciso pero no se adapta bien a dispositivos con diferentes tamaños de pantalla. En pantallas pequeñas, requiere desplazamiento horizontal, y en pantallas grandes, deja espacios en blanco.

Ejemplo de Diseño ancho Fijo:

```html
<div class="fixed-width-container">
  <p>Este es un diseño de ancho fijo.</p>
</div>
```

```css
.fixed-width-container {
  width: 960px; /* Ancho fijo */
  margin: 0 auto;
  background-color: #f0f0f0;
  padding: 20px;
}
```

- Diseño líquido o fluido: Utiliza unidades relativas como porcentajes, permitiendo que el diseño se ajuste al tamaño de la pantalla. Sin

embargo, puede hacer que el contenido se vea demasiado estirado o comprimido en ciertas resoluciones.

**Ejemplo de Diseño Liquido o Fluido**:

```
<div class="fluid-container">
  <p>Este es un diseño líquido o fluido.</p>
</div>
```

```
.fluid-container {
  width: 80%; /* Ancho relativo */
  margin: 0 auto;
  background-color: #e0f7fa;
  padding: 20px;
}
```

- Diseño adaptativo: Ajusta la estructura del diseño en función del ancho del navegador, modificando columnas, ocultando elementos y optimizando la visualización en cualquier dispositivo.

**Ejemplo de Diseño Adaptativo:**

```
<div class="flexible-grid">
  <div class="column">Columna 1</div>
  <div class="column">Columna 2</div>
  <div class="column">Columna 3</div>
</div>
```

```
.flexible-grid {
  display: flex;
  gap: 10px;
}

.column {
  flex: 1; /* Cada columna ocupa una fracción igual del espacio */
  background-color: #ffccbc;
  padding: 20px;
  text-align: center;
}
```

# 3. Técnicas Clave de Diseño Adaptativo

### 3.1 Parrillas Flexibles

Las parrillas flexibles permiten crear diseños fluidos usando porcentajes en lugar de píxeles para las anchuras de las columnas. Este enfoque es crucial para que el diseño se adapte a diferentes dispositivos sin problemas. Por ejemplo, si un diseño de 960px tiene una barra lateral de 200px y un contenido principal de 760px, se convertirían en 20% y 80% respectivamente.

### 3.2 Contenidos Multimedia Adaptativos

Para evitar problemas de desbordamiento en imágenes y videos, se utiliza la propiedad max-width: 100% en CSS, que permite que estos elementos se redimensionen según el contenedor sin superar su tamaño original. Además, es recomendable eliminar los atributos height y width en los elementos <img> para evitar conflictos.

**Ejemplo de max-width: 100% en imágenes:**

```html
<div class="grid-layout">
  <div class="header">Header</div>
  <div class="sidebar">Sidebar</div>
  <div class="main">Main Content</div>
  <div class="footer">Footer</div>
</div>
```

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

# 4. Media Queries

Las media queries de CSS permiten cambiar los estilos según la resolución de la pantalla. Esto es esencial para que el diseño se vea bien en cualquier dispositivo. Las media queries pueden usar operadores como not, and y only y aplicarse a tipos de medios como screen, print, y speech.

**Estrategias de Implementación:**

- Mobile First: Se diseña primero para dispositivos móviles, y luego se adaptan estilos para pantallas más grandes. Este enfoque es más liviano y optimiza el rendimiento móvil.
- Desktop First: El diseño se inicia para pantallas grandes y luego se simplifica para dispositivos más pequeños, añadiendo media queries para ajustar elementos.

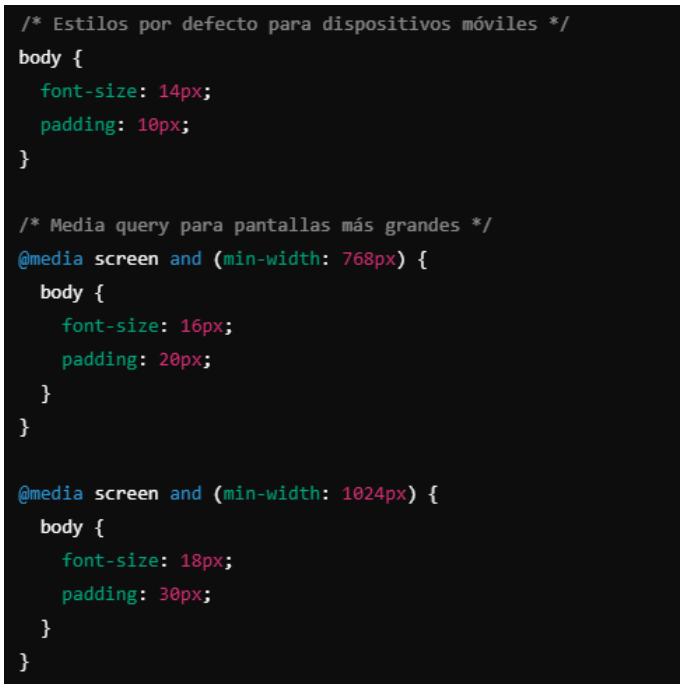
**Ejemplo de Mobile First:**

```css
/* Estilos básicos para móviles */
.container {
  display: block;
  padding: 10px;
  font-size: 14px;
}

/* Estilos para tabletas */
@media screen and (min-width: 768px) {
  .container {
    display: flex;
    font-size: 16px;
  }
}

/* Estilos para escritorios */
@media screen and (min-width: 1024px) {
  .container {
    font-size: 18px;
    padding: 20px;
  }
}
```

**Ejemplo de Mdia Query:**



```css
/* Estilos por defecto para dispositivos móviles */
body {
  font-size: 14px;
  padding: 10px;
}

/* Media query para pantallas más grandes */
@media screen and (min-width: 768px) {
  body {
    font-size: 16px;
    padding: 20px;
  }
}

@media screen and (min-width: 1024px) {
  body {
    font-size: 18px;
    padding: 30px;
  }
}
```

# 5. Modelos de Diseño CSS

### 5.1 Flexbox

Flexbox es un módulo de diseño de CSS3 que simplifica la alineación y distribución de elementos en la página. Está compuesto por un contenedor padre (flex container) y elementos hijos (flex items).

**Propiedades del contenedor Flexbox:**

- display: Define el contenedor como flex o inline-flex.
- flex-direction: Establece la dirección del eje principal (row, row-reverse, column, column-reverse).
- justify-content: Alinea los elementos en el eje principal (flex-start, center, flex-end, space-between, space-around, space-evenly).
- align-items: Alinea los elementos en el eje transversal (flex-start, center, flex-end, stretch).
- flex-wrap: Controla si los elementos permanecen en una sola línea o se dividen en varias (nowrap, wrap, wrap-reverse).

**Ejemplo de Flexbox:**

```html
<div class="flex-container">
  <div class="flex-item">1</div>
  <div class="flex-item">2</div>
  <div class="flex-item">3</div>
</div>
```

```css
.flex-container {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px;
  background-color: #f0f0f0;
}

.flex-item {
  background-color: #4caf50;
  padding: 20px;
  color: white;
  border-radius: 5px;
}
```

**Propiedades de los elementos Flex:**

- order: Controla el orden de los elementos (valor por defecto: 0).
- flex-grow: Define cuánto puede crecer un elemento (valor por defecto: 0).
- flex-shrink: Define cuánto puede reducirse un elemento (valor por defecto: 1).
- flex-basis: Especifica el tamaño inicial de un elemento antes de que se reparta el espacio restante.
- align-self: Sobrescribe la alineación establecida por align-items para un elemento específico.

**Ejemplo de propiedades de elementos Flex:**

```html
<div class="flex-container">
  <div class="flex-item" style="flex-grow: 1;">Item 1 (grow: 1)</div>
  <div class="flex-item" style="flex-grow: 2;">Item 2 (grow: 2)</div>
  <div class="flex-item" style="flex-grow: 1;">Item 3 (grow: 1)</div>
</div>
```

**5.2 CSS Grid**

CSS Grid es un sistema de diseño que permite crear layouts complejos basados en una cuadrícula de filas y columnas. Para usarlo, se define un contenedor con display: grid.

**Propiedades del contenedor Grid:**

- grid-template-columns / rows: Define el número y tamaño de las columnas y filas.
- grid-gap: Establece el espacio entre filas y columnas.
- grid-template-areas: Crea plantillas de diseño con áreas etiquetadas.
- Propiedades de los elementos Grid:

- grid-column / grid-row: Indica la posición de un elemento en la cuadrícula.
- grid-area: Asigna un elemento a una sección específica de la cuadrícula.
-

**Ejemplo de GRID Layout:**

```html
<div class="grid-container">
  <div class="grid-item">A</div>
  <div class="grid-item">B</div>
  <div class="grid-item">C</div>
  <div class="grid-item">D</div>
</div>
```

```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 2fr;
  grid-template-rows: auto;
  grid-gap: 10px;
}

.grid-item {
  background-color: #2196f3;
  padding: 20px;
  color: white;
  text-align: center;
}
```

**Ejemplo de GRID-Template-areas:**

```html
<div class="grid-layout">
  <div class="header">Header</div>
  <div class="sidebar">Sidebar</div>
  <div class="main">Main Content</div>
  <div class="footer">Footer</div>
</div>
```

```css
.grid-layout {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
  grid-template-columns: 1fr 3fr;
  grid-template-rows: auto 1fr auto;
  gap: 10px;
}

.header {
  grid-area: header;
  background-color: #ff9800;
}

.sidebar {
  grid-area: sidebar;
  background-color: #4caf50;
}

.main {
  grid-area: main;
  background-color: #2196f3;
}

.footer {
  grid-area: footer;
  background-color: #f44336;
}
```

# 6. Ejemplos y Usos Combinados

- Uso combinado de media queries con Flexbox y Grid: Se pueden utilizar media queries para cambiar la disposición y las propiedades de los elementos Flex y Grid, garantizando un diseño completamente adaptable.

**Ejemplo de uso combinado Media Queris cn Flexbox Grid:**

```html
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

```css
/* Estilos base usando Flexbox para móviles */
.container {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.item {
  background-color: #4caf50;
  color: white;
  padding: 15px;
  text-align: center;
}

/* Media query para pantallas más grandes usando Grid */
@media screen and (min-width: 600px) {
  .container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
  }
}
```

- Mobile First vs. Desktop First: Elegir entre estas estrategias depende del público objetivo y las estadísticas de uso de dispositivos.

# 7. Buenas Prácticas para el Diseño Adaptativo

- Usar media queries para modificar márgenes, tamaños de fuente y espacios en dispositivos pequeños.
- Empezar con un enfoque Mobile First para garantizar una carga eficiente y escalabilidad.

**Ejemplo diseño que empieza con un enfoque Mobile First puede adaptarse progresivamente a pantallas más grandes:**

```html
<div class="adaptive-content">
  <h1>Contenido Adaptativo</h1>
  <p>Este es un ejemplo de buenas prácticas en el diseño adaptativo.</p>
</div>
```

```css
/* Estilos base para móviles */
.adaptive-content {
  padding: 10px;
  font-size: 16px;
  background-color: #e1f5fe;
}

/* Ajustes para tabletas */
@media screen and (min-width: 768px) {
  .adaptive-content {
    padding: 20px;
    font-size: 18px;
  }
}

/* Ajustes para pantallas de escritorio */
@media screen and (min-width: 1024px) {
  .adaptive-content {
    padding: 30px;
    font-size: 20px;
  }
}
```

- Combinar Flexbox y Grid según las necesidades del proyecto, ya que cada uno tiene ventajas únicas para organizar y alinear elemento