### **Resumen: CSS Avanzado**

CSS avanzado ofrece herramientas poderosas para transformar, animar y mejorar la experiencia visual de una página web. A continuación, se describen sus conceptos principales:

### **Prefijos de Navegadores**

Algunas propiedades CSS3 requieren prefijos para asegurar compatibilidad:

* **-webkit-**: Chrome, Safari, Opera.
* **-moz-**: Firefox.
* **-ms-**: Internet Explorer.

Consulta compatibilidad en: [CSS3 Browser Support](https://www.w3schools.com/cssref/css3_browsersupport.asp).

### **Transformaciones 2D**

Permiten modificar rotación, inclinación, escala y posición de un elemento.

| **Transformación** | **Sintaxis** | **Ejemplo** |
| --- | --- | --- |
| **Rotar** | transform: rotate(grados); | rotate(25deg) |
| **Inclinar** | transform: skew(X, Y); | skew(15deg, 3deg) |
| **Escalar** | transform: scale(X, Y); | scale(1.5, 0.6) |
| **Desplazar** | transform: translate(X, Y); | translate(12px, 19px) |

**Transformaciones combinadas:**  
transform: scale(1.6) skew(10deg) translate(5px) rotate(12deg);

### 

### 

### 

### **Transformaciones 3D**

Añaden profundidad mediante ejes X, Y, y Z.

| **Transformación** | **Sintaxis** |
| --- | --- |
| **Rotar en X** | transform: rotateX(ang); |
| **Rotar en Y** | transform: rotateY(ang); |
| **Rotar en Z** | transform: rotateZ(ang); |

Más información en: [CSS3 3D Transforms](https://www.w3schools.com/css/css3_3dtransforms.asp).

### **Transiciones**

Hacen que los cambios en propiedades CSS sean graduales.

**Propiedades básicas:**  
transition: property duration timing-function delay;

**Ejemplo:**  
button {

transition: background-color 0.3s ease-in-out;

}

**Valores comunes:**

* linear: Cambio constante.
* ease: Cambio acelerado y desacelerado suavemente. (Por defecto)
* ease-in: Comienza lentamente, luego acelera.
* ease-out: Comienza rápido, luego desacelera.
* ease-in-out: Combina ease-in y ease-out.

### 

### 

### 

### **Animaciones**

Permiten cambiar propiedades de un elemento a lo largo del tiempo utilizando fotogramas clave (@keyframes).

**Crear animación:**  
@keyframes mianimacion {

from { left: 0px; background-color: red; }

to { left: 30px; background-color: blue; }

}

1. **Aplicar animación:**  
   #elemento {

animation-name: mianimacion;

animation-duration: 3s;

animation-iteration-count: infinite;

animation-direction: alternate;

}

1. **Opciones avanzadas:**
   * animation-delay: Retraso antes de iniciar la animación.
   * animation-timing-function: Suavizado (ease, linear, etc.).
   * animation-fill-mode: Controla estilos antes/después de la animación (none, forwards, backwards, both).

**Ejemplo Completo:**

@keyframes mover {

0% { left: 0px; background-color: red; }

50% { left: 10px; background-color: yellow; }

100% { left: 30px; background-color: blue; }

}

#miElemento {

position: relative;

animation: mover 2s infinite alternate ease-in-out;

}

### 

### 

### **Valores Posibles de animation-direction**

* **normal (por defecto):**La animación avanza de inicio a fin en cada iteración.  
  Ejemplo: 0% → 100%.
* **reverse:**La animación comienza desde el final hacia el inicio en cada iteración.  
  Ejemplo: 100% → 0%.
* **alternate:**Alterna entre avanzar (normal) y retroceder (reverse) en cada iteración.  
  Ejemplo:
  + Iteración 1: 0% → 100%.
  + Iteración 2: 100% → 0%.
* **alternate-reverse:**Alterna como alternate, pero empieza desde el final en la primera iteración.  
  Ejemplo:
  + Iteración 1: 100% → 0%.
  + Iteración 2: 0% → 100%.

### **Display: flex (Flexbox)**

Flexbox es un modelo de diseño en CSS que facilita la alineación y distribución de elementos dentro de un contenedor.

### **Propiedades**

* + Activa el modelo flexbox para el contenedor.
  + Sus elementos hijos se convierten en elementos flexibles.

1. **flex-direction**
   * Define la **dirección del eje principal** (horizontal o vertical).
   * Valores:
     + row (por defecto): Elementos en fila horizontal.
     + row-reverse: Fila horizontal, orden inverso.
     + column: Elementos en columna vertical.
     + column-reverse: Columna vertical, orden inverso.
2. **justify-content**
   * Alinea los elementos a lo largo del eje principal.
   * Valores:
     + flex-start (por defecto): Hacia el inicio.
     + flex-end: Hacia el final.
     + center: En el centro.
     + space-between: Espaciados uniformemente, extremos pegados.
     + space-around: Espaciados uniformes con márgenes.
     + space-evenly: Espacios iguales entre y alrededor.
3. **align-items**
   * Alinea los elementos a lo largo del eje cruzado (perpendicular al principal).
   * Valores:
     + stretch (por defecto): Elementos estirados para ocupar el eje cruzado.
     + flex-start: Hacia el inicio del eje cruzado.
     + flex-end: Hacia el final.
     + center: Centrados en el eje cruzado.
     + baseline: Alineados por la línea base del texto.
4. **flex-wrap**
   * Define si los elementos deben ajustarse en varias líneas si no caben en una.
   * Valores:
     + nowrap (por defecto): No ajusta, todos en una línea.
     + wrap: Ajusta a una nueva línea si es necesario.
     + wrap-reverse: Ajusta en orden inverso.
5. **align-content**
   * Alinea múltiples líneas de elementos (cuando hay un ajuste con flex-wrap).
   * Valores:
     + stretch (por defecto): Las líneas se estiran para ocupar el espacio.
     + flex-start: Hacia el inicio.
     + flex-end: Hacia el final.
     + center: Centradas.
     + space-between: Espaciadas uniformemente.
     + space-around: Espaciadas con márgenes.

### **Responsive Web Design: Apuntes Simples**

El diseño **responsive** asegura que una página web se adapte correctamente a cualquier dispositivo, como **desktop**, **tablet**, o **móvil**, usando **HTML** y **CSS**.

### 

### 

### 

### 

### **Responsive Design**

#### **1. Viewport**

Añadir esta línea en el <head> asegura que la página se adapta al ancho del dispositivo:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

#### **2. Layout**

Limitar el ancho del contenedor principal:  
#main { max-width: 960px; min-width: 320px; margin: 0 auto; }

* Usar **porcentajes** para anchos y márgenes para mayor flexibilidad.
* Hacer cálculos combinando **pixeles y porcentajes** con calc().
* Usar box-sizing: border-box para evitar que márgenes y bordes afecten el tamaño total.

#### **3. Media Queries**

Permiten aplicar reglas CSS específicas según el tamaño de pantalla.  
Sintaxis básica:

@media only screen and (max-width: 500px) {

/\* Reglas para pantallas de 500px o menos \*/

}

@media only screen and (min-width: 800px) {

/\* Reglas para pantallas de 800px o más \*/

}

@media only screen and (min-width: 500px) and (max-width: 800px) {

/\* Reglas entre 500px y 800px \*/

}

#### **4. Acciones Comunes en Media Queries**

**Ocultar elementos:**  
display: none;

* **Hacer elementos al 100% de ancho en móviles:**
  + Útil para elementos flotantes o alineados que necesitan ajustarse a una columna.
* **Ajustar márgenes y paddings** para mejorar la visualización en pantallas pequeñas.

#### **5. Landscape**

Detecta si el dispositivo está en modo horizontal:

@media only screen and (orientation: landscape) {

body { background-color: lightblue; }

}

### **Consejos Generales**

1. Mantén el HTML **limpio** y sin estilos en las etiquetas.

Usa **imágenes escalables**:  
.miimagen { max-width: 100%; height: auto; }

### **Tamaños Comunes de Dispositivos**

* **Móviles verticales:** 320px–400px.
* **Móviles horizontales y tablets:** 400px–800px.
* **Sobremesa:** 800px o más.
* **display: grid;**Activa el modelo de cuadrícula en el contenedor.
* **grid-template-columns y grid-template-rows**Define el tamaño de las columnas y filas:
  + **Unidades relativas:**
    - 1fr para dividir el espacio proporcionalmente.
    - Ejemplo: grid-template-columns: 1fr 2fr;
  + **Unidades absolutas:**
    - Ejemplo: grid-template-columns: 150px 200px;
  + **Mixtas:**
    - Ejemplo: grid-template-rows: 100px auto 50%;

**gap**Espacio entre filas y columnas:  
gap: 10px; /\* Uniforme \*/

row-gap: 10px; /\* Entre filas \*/

column-gap: 20px; /\* Entre columnas \*/

* **justify-content y align-content**Ajustan la distribución dentro del contenedor:
  + justify-content: space-evenly; (horizontal)
  + align-content: center; (vertical)

### **Grid**

1. **grid-column y grid-row**Controlan el inicio y final de un elemento:

**Ejemplo básico:**  
grid-column: 1 / 3; /\* Ocupará de la columna 1 a la 3 \*/

grid-row: 2 / 4; /\* Ocupará de la fila 2 a la 4 \*/

**Extensión total:**  
grid-column: 1 / -1; /\* Ocupará todo el ancho \*/

1. **justify-self y align-self**Ajustan la posición de un elemento individual:
   * justify-self: center; (horizontal)
   * align-self: end; (vertical)