Capítulo 3

Unidades de medida

Unidades de medida

- Las medidas en **CSS** se emplean para definir la *altura, anchura y márgenes de los elementos* y para establecer el tamaño de letra del texto.

- Todas las medidas se indican como un valor numérico entero o decimal seguido de una unidad de medida.

Unidades de medida

- **CSS** divide todas las unidades de medida en dos grupos: *absolutas* y *relativas*.

- Las medidas relativas definen su valor *en relación con otra medida*.

- Las unidades absolutas *establecen de forma completa el valor de una medida*.

Unidades relativas

- La unidades relativas son más **flexibles** que las unidades absolutas porque se adaptan más fácilmente a los diferentes medios.

- A continuación se muestra la lista de unidades de medida relativas:
  - **em**, relativa respecto de la anchura de la letra **M** del tipo de letra que se esté utilizando. Lo toma del font-size.
  - **ex**, relativa respecto de la altura de la letra *x* del tipo de letra que se esté utilizando.
  - **px**, (píxel) relativa respecto de la pantalla del usuario.

Unidades relativas

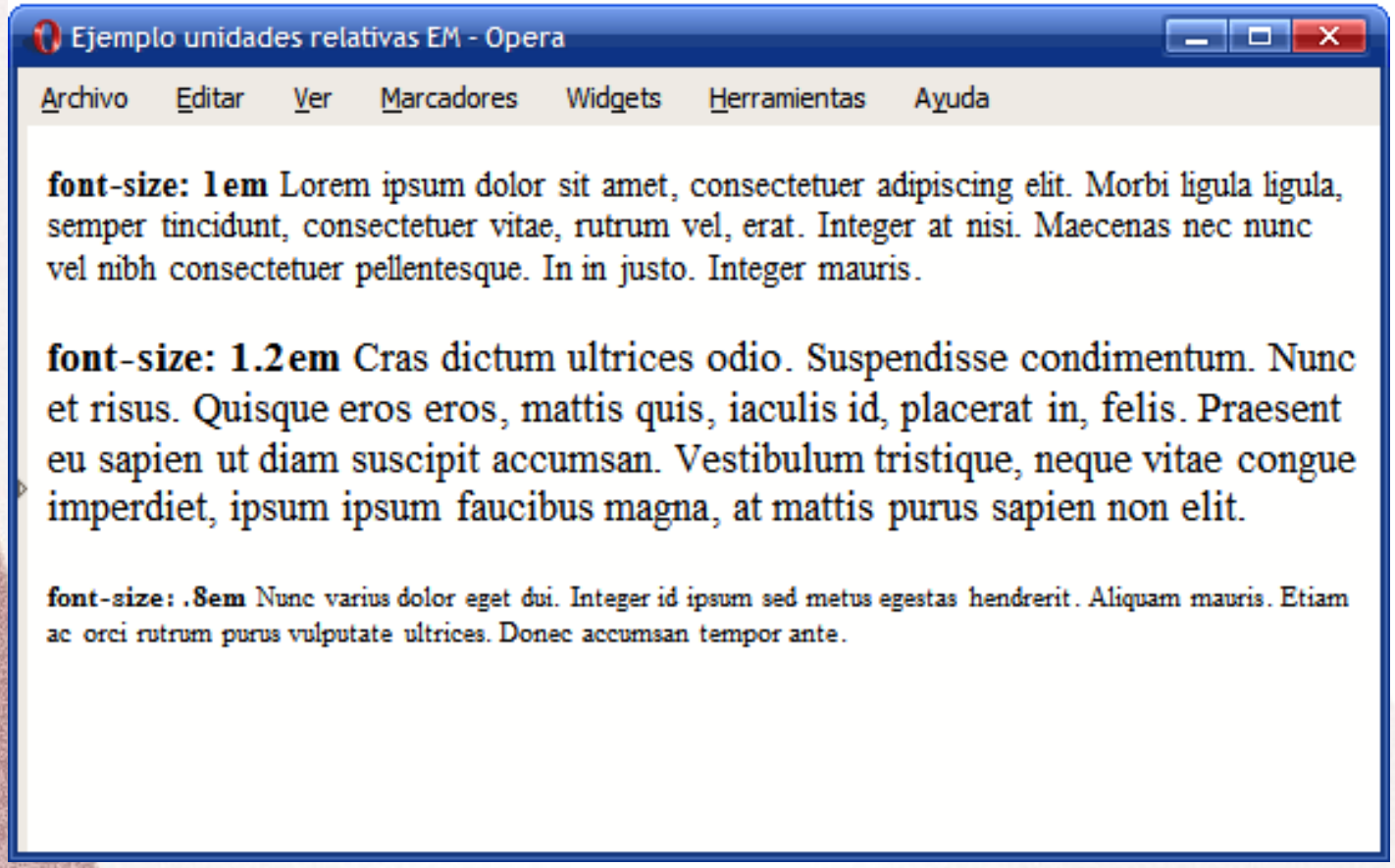
- La propiedad **font-size** permite establecer el tamaño de letra del texto de un elemento.

- Si usamos unidades relativas, la referencia es el tamaño de letra de su elemento padre.

- Si no se indica de forma explícita un valor para el tamaño de letra del elemento **<html>**, la referencia es el *tamaño de letra por defecto del navegador*.

Unidades relativas - em

- *Ejemplo* que muestra el uso de la unidad **em**:



Unidades relativas - combinación

- Las distintas unidades se pueden mezclar entre los diferentes elementos de una misma página, ejemplo:

```
body { font-size: 10px; }
h1 { font-size: 2.5em; }
```

Unidades relativas - combinación

- De hecho, se suele realizar una configuración como la siguiente, útil para diseño adaptable:

```
html {
  font-size: 100%; /* 100% = 16px */
}
p {
  font-size: 1em; /* 1em = 16px */
}
h1 {
  font-size: 2em; /* 2em = 32px */
}
```

Unidades relativas - em

- El 100% que hemos puesto en el html equivale al tamaño de fuente de la configuración del navegador, cuyo valor predeterminado suele ser de 16px.
- Algunos desarrolladores prefieren utilizar un font-size de 62.5% para <html>. De este modo reducen el valor predeterminado de 16px hasta 10px y así pueden realizar los cálculos en múltiplos de 10, más simple y sencillo; por ejemplo, 1em = 10px, 1.2em = 12px, etc.

Unidades relativas - problema

- ¿Cuál es el problema con estos em?
- El em depende del tamaño de la fuente del contenedor, o del tamaño de fuente declarado en alguno de los contenedores padre de éste. Ejemplo:

```
html {
 font-size: 100%; /* 100% = 16px */
}
.article {
  font-size: 2em; /* Ahora 1em = 32px en elementos anidados */
}
p {
  /* Se hereda el font-size, así que aquí 1em = 32px */
  font-size: 0.5em; /* 0.5 × 32px = 16px */
}
```

Unidades relativas – em para padding y margin

- Como em es relativa al font-size del elemento, será muy útil en propiedades que requieran mantener la proporcionalidad con el font-size del elemento; por ejemplo, es muy habitual en margin y padding.

```
html {
font-size: 100%; /* 100% = 16px */
}
h1 {
  font-size: 2.197em; /* 2.197 × 16px = 35px, así que para h1 1em = 2.197 */
  margin-top: 0.59em; /* 0.59 em = 20px, ya que 35 x 0.59 = 20  */
  margin-bottom: 0.59em;
}
```

Unidades relativas – em para padding y margin

- Ojo en el ejemplo anterior, pues el font-size de h1 se hereda de html pero los padding y margin del nuevo font-size.
- Para h1, el valor de font-size se hereda desde html, que era de 16px, y se establece en 2.197em, que equivale a 35px.
- En ese momento, el font-size del elemento sería de 35px, es decir, para el resto de propiedades en h1, 1em sería igual a 35px, ya que, si recordamos la definición de em, 1em es igual al tamaño de fuente definido para el elemento actual.
- En otras palabras, las propiedades definidas en em cambian con del font-size del elemento.

Unidades relativas rem

- Rem, que viene de Root EM, no funciona de manera relativa a su contenedor, sino de manera relativa al tamaño definido en la raíz.
- La regla del "target/context" sigue valiendo perfectamente, pero puedo hacer la cuenta siempre con respecto al tamaño definido en la raíz.
- Si en el elemento HTML teníamos 16px de tamaño y quiero 24px en el H1, tengo que calcular 24/16 = 1.5rem.
- Si luego en el enlace que había dentro del H1 quiero que sea 12px, tengo que calcular como context 16px (que era el equivalente al 100% definido como tamaño de la raíz) y no el tamaño del H1. En este caso sería 12/16 = 0.75rem.

Unidades relativas rem

- Rem no funciona de manera relativa a su contenedor, sino de manera relativa al tamaño definido en la raíz.
- La regla del "target/context" sigue valiendo perfectamente, pero puedo hacer la cuenta siempre con respecto al tamaño definido en la raíz.
- Si en el elemento HTML teníamos 16px de tamaño y quiero 24px en el H1, tengo que calcular 24/16 = 1.5rem.
- Si luego en el enlace que había dentro del H1 quiero que sea 12px, tengo que calcular como context 16px (que era el equivalente al 100% definido como tamaño de la raíz) y no el tamaño del H1. En este caso sería 12/16 = 0.75rem.

Unidades relativas - rem

- Ejemplo:

```
html {
  font-size: 100%; /* 100% = 16px */
}
h1 {
  font-size: 2.197rem; /* 2.197 × 16px = 35px */
  margin-top: 1.25rem; /* 1.25rem = 20px, ya que 1.25 × 16 = 20 */
  margin-bottom: 1.25rem;
}
```

- Debido a que 1rem es constante a lo largo del documento, los cálculos a realizar son más sencillos que con em.

Unidades relativas - combinando em y rem

- Para un selector CSS dado, las propiedades definidas en em cambian con el font-size del elemento actual, las propiedades definidas en rem siempre cambian con el font-size del elemento <html>.
- Hay quien recomienda:
  - Utiliza em para propiedades que queremos que cambien con el font-size del elemento
  - Utiliza rem para todo lo demás
- Hay artículos y debates en torno a esta cuestión:

  https://zellwk.com/blog/rem-vs-em/
- Ejemplo para hacer pruebas:

  https://codepen.io/CybMeta/pen/RGZbpz

Unidades relativas

- Si el usuario tiene problemas de visión y aumenta el tamaño de letra en su navegador, las proporciones se mantendrán.

- Las *unidades relativas* permiten mantener las proporciones del diseño independientemente del tamaño de letra por defecto del navegador del usuario.

Unidades relativas - ex

- El funcionamiento de la unidad **ex** es idéntico a **em**, salvo que en este caso, la referencia es la *altura* de la letra x minúscula.

- Las medidas indicadas en píxel también se consideran relativas, ya que el aspecto de los elementos dependerá de la resolución del dispositivo en el que se visualiza el documento HTML.

Unidades absolutas

- Las *unidades absolutas* definen las medidas de forma completa, ya que sus valores reales son directamente los valores indicados.

- La *lista completa de unidades absolutas* definidas por **CSS** y su significado:
  - in, pulgadas (1 pulgada son 2.54 centímetros)
  - cm, centímetros
  - mm, milímetros
  - pt, puntos (1 punto equivale a 1 pulgada/72, es decir, unos 0.35 milímetros)
  - pc, picas (1 pica equivale a 12 puntos, es decir, unos 4.23 milímetros)
  - px, píxeles, se considera relativa pero en cierto modo podemos considerarla si no absoluta, sí estática.

Unidades absolutas

- Ejemplos de utilización de unidades absolutas:

```
body { margin: 0.5in; }
h1 { line-height: 2cm; }
p { word-spacing: 4mm; }
a { font-size: 12pt }
span { font-size: 1pc }
```

- *Los valores indicados son directamente los valores que se utilizan*, sin necesidad de calcular los valores reales en función de otras referencias.

Unidades absolutas

- De todas las unidades absolutas, la única que se utiliza con cierta frecuencia es la de los **puntos** (***pt***).

- *Se trata de la unidad preferida* para indicar el tamaño de letra del texto para los documentos que se van a imprimir, es decir, para el medio ***print*** de **CSS**.

Porcentajes

- **CSS** define otra unidad de medida relativa basada en los *porcentajes*.

- Un *porcentaje* está formado por un valor numérico seguido del símbolo % y siempre está referenciado a otra medida.

Porcentajes

- Los porcentajes se pueden utilizar por ejemplo para establecer el valor del tamaño de letra de los elementos:

```
body { font-size: 1em; }
h1 { font-size: 200%; }
h2 { font-size: 150%; }
```

- Los tamaños establecidos para **<h1>** y **<h2>** son equivalentes a 2em y 1.5em.

Porcentajes

- Los porcentajes también se utilizan para establecer la anchura de los elementos:

```
div#contenido { width: 600px; }
div.principal { width: 80%; }

<div id="contenido">
<div class="principal">
...
</div>
</div>
```

Recomendaciones

- En general, se recomienda el uso de *unidades relativas* siempre que sea posible,
  - *mejora la accesibilidad* de la página y
  - permite que los documentos se adapten fácilmente a cualquier medio y dispositivo.

Recomendaciones

- Normalmente se utilizan **píxel** y **porcentajes** para definir el layout del documento (la *anchura de las columnas* y *elementos* de las páginas)
- **em/rem** y **porcentajes** para el *tamaño de letra* de los textos.

Recomendaciones

- El siguiente ejemplo muestra el primer caso:

```
div { font-size: 0.9em; }

<div>
    <p>Texto 1</p>
    <div>
        <p>Texto 2</p>
        <div>
            <p>Texto 3</p>
        </div>
    </div>
</div>
```

Recomendaciones

Texto 1

Texto 2

Texto 3

Colores

- Los colores en **CSS** se pueden indicar de cinco formas diferentes:
  - palabras clave,
  - colores del sistema,
  - **RGB** hexadecimal,
  - **RGB** numérico y
  - **RGB** porcentual.

- El método más habitual es el del ***RGB hexadecimal***.

Palabras clave

- **CSS** define 17 palabras clave para referirse a los colores básicos.

aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow

- Se trata de una *gama de colores muy limitada*.

Palabras clave

RGB decimal

- El modelo **RGB** consiste en *definir un color indicando la cantidad de color rojo*, *verde* y *azul* que se debe *mezclar* para obtener ese color.

- Este es un modelo de tipo "*aditivo*", ya que los colores se obtienen sumando sus componentes.

RGB decimal

- Si todas las componentes valen 0, el color creado es el negro y si todas las componentes toman su valor máximo, el color obtenido es el blanco.

- En **CSS**, las componentes de los colores definidos mediante **RGB** decimal pueden tomar valores entre 0 y 255.

- *Ejemplo*:

```
p { color: rgb(71, 98, 176); }
```

RGB decimal

- La *sintaxis* que se utiliza para indicar los colores es **rgb()** y entre paréntesis se indican las tres componentes **RGB**, en ese mismo orden y separadas por comas.

RGB porcentual

- Otra forma de indicar las componentes **RGB** de un color es mediante un *porcentaje*.

- La única diferencia en este caso es que el valor de las componentes **RGB** puede tomar valores entre 0% y 100%.

- El mismo color del ejemplo anterior se puede representar de forma porcentual:

```
p { color: rgb(27%, 38%, 69%); }
```

RGB porcentual

- Al igual que sucede con el **RGB** decimal, si se indica un valor inferior a 0%, se transforma automáticamente en 0% y si se indica un valor superior a 100%, se trunca su valor a 100%.

RGB hexadecimal

- Aunque es el método *más complicado* para indicar los colores, se trata del método más utilizado con mucha diferencia.

- Para entender el modelo **RGB** hexadecimal, en primer lugar es preciso introducir un concepto matemático llamado *sistema numérico hexadecimal*.

RGB hexadecimal

- Para definir un color en **CSS** con **RGB** hexadecimal se realizan los siguientes pasos:

1. Se determinan las componentes RGB del color original, por ejemplo: R = 71, G = 98, B = 176.

2. El valor numérico de cada componente se transforma al sistema numérico hexadecimal. En el ejemplo anterior, el valor hexadecimal de cada componente es: R = 47, G = 62, B = B0.

3. Para obtener el color completo en formato RGB hexadecimal, se concatenan los valores de las componentes **RGB** en ese orden y se les añade el prefijo #. De esta forma, el color del ejemplo anterior es **#4762B0** en formato *RGB hexadecimal*.

RGB hexadecimal

- Ejemplo:

```
p { color: #4762B0; }
```

RGB hexadecimal

- En el siguiente ejemplo:

```
body { background-color: #FFF; color: #000; }
h1, h2, h3, h4, h5, h6 { color: #C00; }
```

URLs

- Cuando una hoja de estilo debe referenciar URLs o vínculos a otros archivos, casi siempre se trata de gráficos que se vinculan como fondo a un elemento.
- Si queremos integrar una imagen de fondo a nuestra página:

```
body {
    background-image: url(../images/imagen.gif);
}
```