educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 2 - Java Web Developer

Módulo 14

Medidas de longitud

Trabajando con em

Las unidades **`em`** no han sido creadas por CSS, sino que llevan décadas utilizándose en el campo de la tipografía. Aunque no es una definición exacta, la unidad **`1em` equivale a la anchura de la letra M (eme mayúscula) del tipo y tamaño de letra del elemento**.

La unidad `em` hace referencia al tamaño en puntos (pt) de la fuente que se está utilizando.

Si se utiliza una tipografía de **`12 pt`**, **`1 em`** equivale a **`12 puntos`**.

PX — Para tamaños y espacios fijos.

vs

EM — Depende del elemento padre.

vs

REM — Fácilmente escalable en el diseño responsivo.

Como se trata de una **unidad de medida relativa,** es necesario realizar un cálculo matemático para determinar el tamaño en **em**.

La unidad de medida em siempre hace referencia al tamaño tipográfico del elemento padre, ya sea este un contenedor o el elemento padre por excelencia el **body**.

Todos los navegadores muestran por defecto el texto (salvo los enunciados) con un tamaño de letra de 16 px. Si realizamos la operación mencionada 1 em equivale entonces a 16 px.

Los h1 son, por defecto, el doble de la medida base, por eso, si lo analizamos con el inspector de elementos, veremos la siguiente regla predeterminada.

```
h1 {
        display: block;
        font-size: 2em;
        margin-block-start: 0.67em;
        margin-block-end: 0.67em;
        margin-inline-start: 0px;
        margin-inline-end: 0px;
        font-weight: bold;
        }
```

Em puede implementarse a cualquier tipo de propiedad que admita medidas de longitud. En el ejemplo debajo se ha trabajado el valor de un margen en referencia al tamaño tipográfico.

```
p {font-size: 32px; margin: 0.5em;}
```

Dado que nuestros párrafos poseen **32 px** de `font-size,` entonces **`0.5 em`** de margen será equivalente a la mitad, por tanto, el margen tiene como resultado **16 px.**

El valor **`0.5 em`** se interpreta como "la mitad del tamaño de letra del elemento", ya que se debe multiplicar por **0.5 su tamaño de letra (`32 px x 0.5 = 16 px`).**

De la misma forma, si se quiere mostrar un margen de **`8 px`** de anchura, debería utilizarse el valor **`0.25 em`**, ya que **`32 px x 0.25 = 8 px`**.

La gran ventaja de **las unidades relativas es que siempre mantienen las proporciones del diseño** de la página. Establecer el margen de un elemento con el valor `1 em` equivale a indicar que "el margen del elemento debe ser del mismo tamaño que su letra y debe cambiar proporcionalmente".

En efecto, si el tamaño de letra de un elemento aumenta hasta un valor enorme, su margen de `1em` también será enorme. Si su **t**amaño de letra se reduce hasta un valor diminuto, el margen de `1em` también será diminuto.

Unidades ex

Existe otra medida ya no utilizada en CSS llamada ***ex***. La unidad **ex** funciona igual a **`em`**, pero en este caso la referencia es la altura de la letra **x (equis minúscula)**, siendo su valor aproximadamente **la mitad que el de la unidad `em`**.

Las unidades de medida se pueden mezclar **en los diferentes elementos de una misma página**, como en el siguiente ejemplo:

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

En primer lugar, se establece un tamaño de letra base de **`10 px`** para toda la página.

A continuación, se asigna un tamaño de **`2.5 em`** al elemento **`<h1>`**, por lo que su tamaño de letra real será de **`2.5 x 10px = 25px`**.

Introducción al trabajo con rem

`rem` es una unidad de medida que posee una diferencia muy importante con **`em`**, **no hereda desde su elemento padre, sino desde el elemento raíz del documento, es decir desde la etiqueta `<html>`**, de ahí viene su nombre ***root em***.

Veámoslo con un ejemplo: en el caso de **`em`**, si un contenedor posee **`2em`** de `font-size` y a su vez un `h1` contenido tiene en el css ese mismo tamaño tipográfico, el resultado final será de **`4em`**, dado que los valores se multiplican. En cambio, `rem` no duplica su valor, puesto que siempre toma al *root* como referencia.

Es importante entender esta diferencia para poder dejar elementos que no cambien su tamaño en base al cambio de tamaño base del elemento padre y así controlar en ambos casos la adaptabilidad de los elementos.

Ambas unidades de medida son necesarias al momento de trabajar en la **creación de proyectos responsivos**, permitiéndonos trabajar de forma más cómoda y eficiente.

Trabajar con porcentajes

El porcentaje también es una unidad de medida relativa, aunque por su importancia CSS la trata de forma separada a `em` y `px`.

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

Cada una de las propiedades de **CSS** que permiten indicar como valor un porcentaje, define el valor al que hace referencia **ese porcentaje**.

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

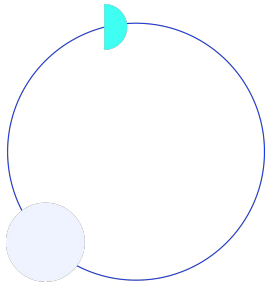
```
body {font-size: 1em;}
h1 {font-size: 200%;}
h2 {font-size: 150%;}
```

El uso más común de los porcentajes es establecer la anchura de los elementos:

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

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

En el ejemplo, la referencia del **valor 80%** es la anchura de su elemento padre. Por tanto, el elemento ** ** cuyo atributo class vale principal tiene una anchura de **80% x 600px = 480px**.



Las unidades de medida se pueden mezclar **en los diferentes elementos de una misma página**, como en el siguiente ejemplo:

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

En primer lugar, se establece un tamaño de letra base de **`10 px`** para toda la página.

A continuación, se asigna un tamaño de **`2.5 em`** al elemento **`<h1>`**, por lo que su tamaño de letra real será de **`2.5 x 10px = 25px`**.

Como se vio en los capítulos anteriores, el valor de la mayoría de propiedades CSS se hereda de **padres a hijos**. Así por ejemplo, si se establece el tamaño de letra al elemento **`<body>`**, todos los elementos de la página tendrán el mismo tamaño de letra, salvo que indique otro valor.

Sin embargo, el **valor de las medidas relativas** no se hereda directamente, sino que se hereda su valor real una **vez calculado**.

El siguiente ejemplo muestra este comportamiento:

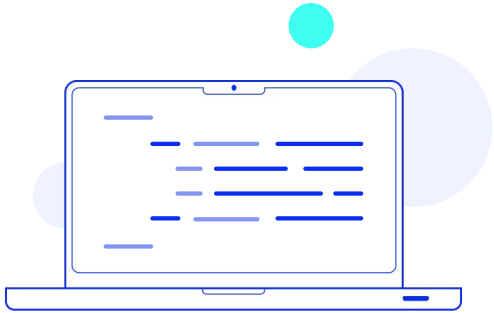
```
body {font-size: 12px; text-indent: 3em;}
h1 {font-size: 15px;}
```

La propiedad **`text-indent`**, se utiliza para tabular la primera línea de un texto. El elemento **`<body>`** define un valor para esta propiedad, pero el elemento **`<h1>`** no lo hace, por lo que heredará el valor de su elemento padre. Sin embargo, el valor heredado no es **`3em`**, sino **`36px`**.

Si se heredará el valor **`3em`**, al multiplicarlo por el valor de **`font-size del elemento <h1> (que vale 15px)`** el resultado sería **`3em x 15px = 45px`**.

No obstante, como se ha comentado, los valores que se heredan no son los relativos, sino los valores ya calculados. Por lo tanto, en primer lugar se calcula el valor real de **`3em`** para el elemento **`<body>: 3em x 12px = 36px`**.

Una vez calculado el valor real, este es el valor que se hereda para el **resto de elementos**.



Trabajar con porcentajes: imágenes responsivas y elementos multimedia

En el caso de las imágenes y en general cualquier elemento multimedia, el uso de **%** combinado con propiedades como **max-width** nos permiten trabajar de manera responsiva.

Un recurso común es **implementar la propiedad max-width para que los elementos multimedia nunca excedan su tamaño original,** y así no pixelar los mismos ni deformarlos.

Evitar el uso de height también es fundamental para que estos elementos no pierdan su proporción.

Ejemplos:

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

```
video {max-width: 100%;}
```

Imágenes responsivas: srcset

Si bien las imágenes y los elementos de multimedia en general, utilizan un modelo más fluido que responsivo de trabajo, usando principalmente porcentajes, como mencionamos antes, no siempre esto representa la solución. Por eso, **la posibilidad de cambiar la imagen en diferentes medios es un recurso interesante.**

Esto se puede lograr a través del atributo **`srcset`**.

```
<img srcset="imagenes/imagen-1.png 300w,imagenes/imagen-2.
png 1000w" alt="Servicios Esenciales">
```

Al trabajar con **srcset** debemos tener en cuenta que si lo trabajamos de la forma anterior, dejamos que el navegador elija según la resolución de pantalla del dispositivo y eso puede generar resultados no deseados.

Para evitar el control del navegador por sobre nuestra decisión podemos sumar datos más específicos para decidir específicamente cuándo se mostrará una imagen u otra.

De todas formas sigue siendo compleja la elaboración de la etiqueta y el atributo dado que depende del radio de pantalla, por lo tanto, a veces, es una opción demasiado elaborada para trabajar de forma cotidiana en procesos de maquetación.

```
<img  srcset="imagenes/imagen-1.png 256w, imagenes/imagen-2.png 1024w"
sizes="(max-width: 768px) 256px, (max-width: 1000px) 1024px " >
```

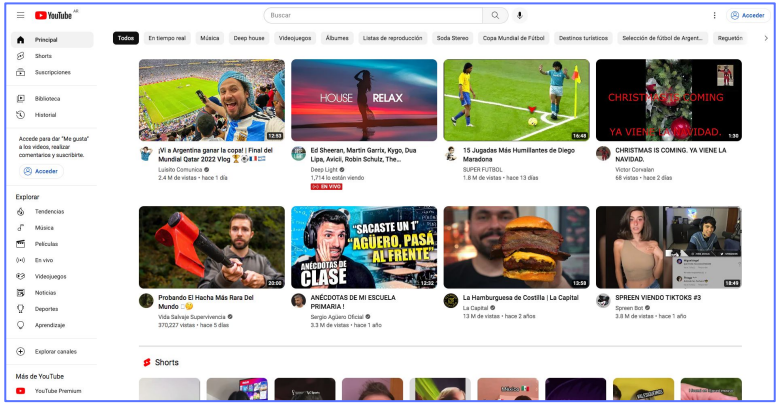
Uso de picture

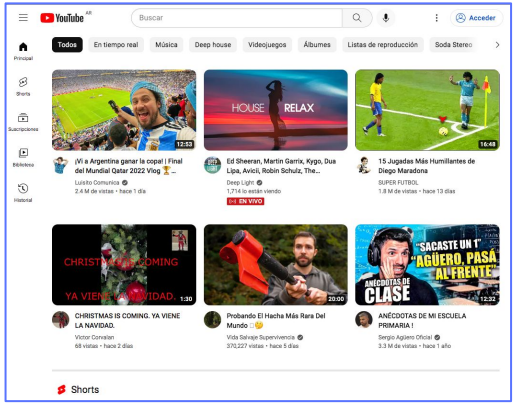
Existe una manera más sencilla para trabajar con imágenes de modo responsivo, usar **`picture`**, que **permite detectar diferentes `breakpoints` y elegir la imagen deseada.**

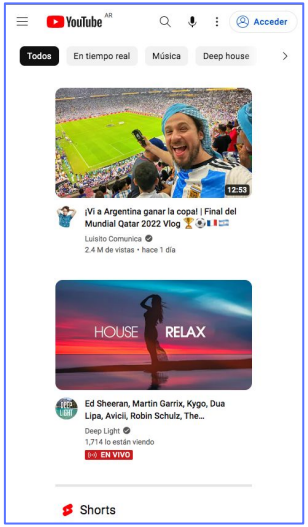
En el ejemplo debajo, en pantallas de **768 px** o inferiores se seleccionará *imagen-1.png*, en pantallas entre **769 px** y **1024px** *imagen-2.png*, y en pantallas superiores a **1024px** elegir la imagen por defecto, que es la *imagen-3.png*.

```
<picture>
<source media="(max-width: 768px)" srcset="imagenes/imagen-1.png">
<source media="(max-width: 1024px)" srcset="imagenes/imagen-2.png">
<img src="imagenes/imagen-3.png" alt="imagen">
</picture>
```

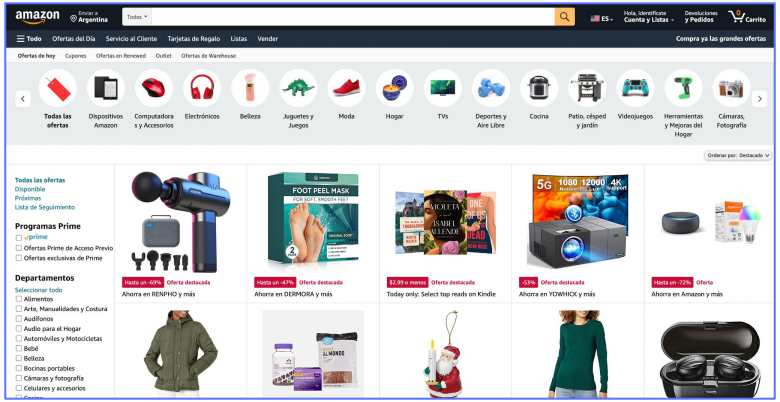
Ejemplo de diseño responsivo en sitios populares

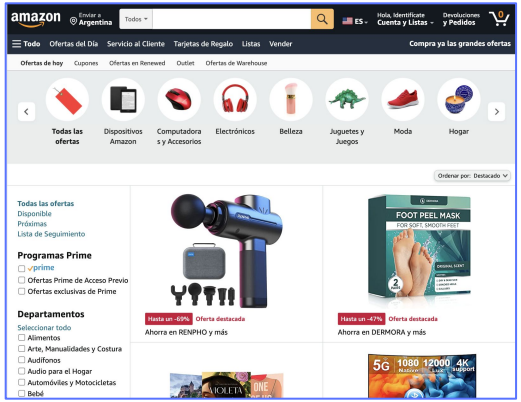




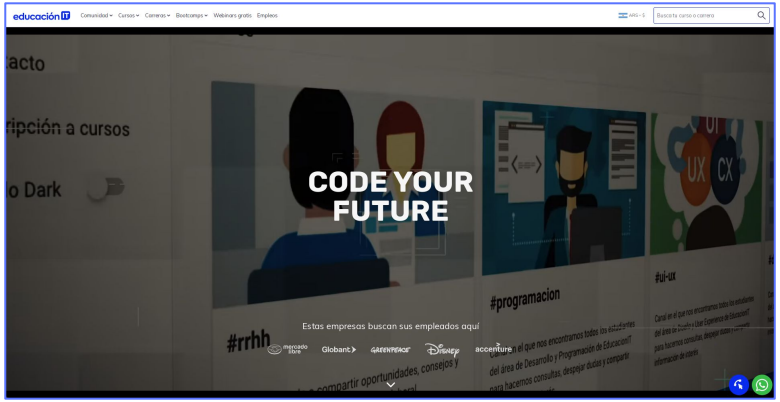


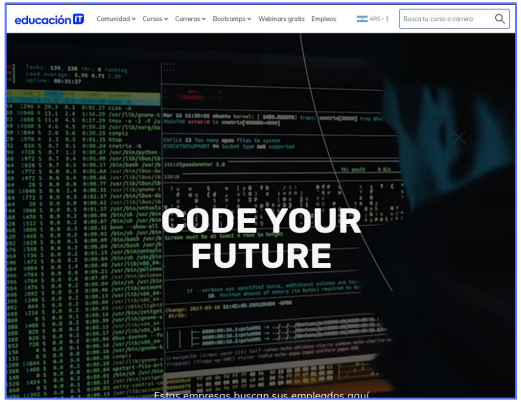
Ejemplo de diseño responsivo en sitios populares

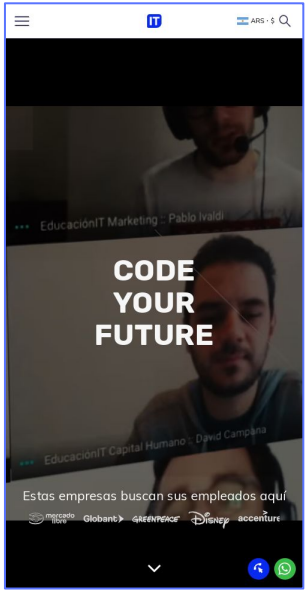




Ejemplo de diseño responsivo en sitios populares







Conclusiones

Debemos siempre priorizar nuestro público y el diseño elegido. Si trabajamos correctamente nada debería generar el sacrificar esta línea en pos de lograr una interfaz responsiva.

Entender qué busca, usa y quiere nuestro usuario es el punto de partida, conocer luego las herramientas y cómo utilizarlas desde el maquetado: HTML y CSS completan el camino hacia una verdadera interfaz responsiva.

Nota: recomendamos utilizar search.google.com/test/mobile-friendly para medir cuán óptima es tu interfaz para dispositivos móviles.

¡Sigamos trabajando!