Programador Web – Nivel 1

Unidad 6: Desarrollo web multiplataforma

Contacto: info@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 6: Desarrollo web multiplataforma

Propiedades de texto

Objetivos

Que el alumno logre:

- Manejar los nuevos elementos incorporados a CSS3 para crear sitios multiplataforma.

Propiedades de Texto

En el proceso de desarrollo del diseño web, una de las grandes limitaciones con las que nos cruzamos tienen que ver con las posibilidades de trabajar con bloques de texto.

CSS3 hace incapié en este problema e implementa algunas propiedades específicas para el manejo de texto, como ser la posibilidad de utilizar cualquier familia tipográfica, de separar el texto en columnas o de cortar largas palabras de texto.

PROPIEDAD @FONT-FACE

@font-face aparece en el nivel 2 de CSS, pero se asienta como estándar en el nivel 3 de CSS.

Esta propiedad básicamente nos permite utilizar en nuestra web cualquier familia tipográfica, sin importar si el usuario la tiene o no instalada en su pc.

Esto se debe a que a través de este estilo, lo que hacemos es indicarle al navegador que no busque la tipografía en la máquina del usuario, sino que lo haga en nuestro servidor.

La implementación de *@font-face* es la siguiente:

```
@font-face{
  font-family:<nombre_fuente>; /*El nombre con el que nos referiremos a la fuente*/
  src: <source>[<ruta de acceso a la fuente>],
/*La ruta de donde cargamos el archivo del tipo*/
  [format:<formato>];
/*Formato de la fuente*/
}
```

Veamos un ejemplo de implementación:

```
@font-face{
        font-family: 'dospuntocero';
        src:url('fuente/duepuntozero_regular.ttf') format('truetype');
    }
    #ejemplo{
```

```
    font-family: 'dospuntocero';
}
```

Podemos visualizar este ejemplo en fuente.html

Una cuestión que provoca algunos problemas con el uso de esta característica es que aún no existe un estándar definitivo de fuentes aceptado por todos los navegadores y plataformas. Po esta razón, y para lograr mayor compatibilidad, hay que incluir las fuentes en diferentes formatos, para que los usuarios puedan verlas correctamente.

Por ejemplo:

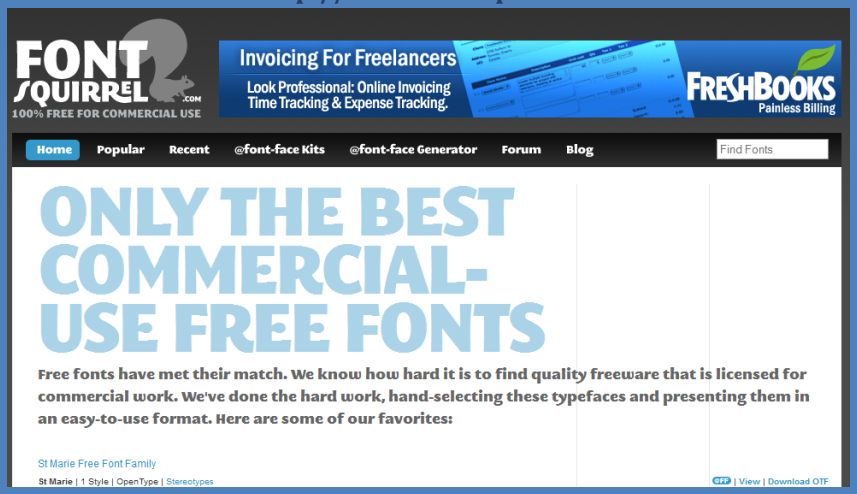
```
src:  url('Delicious-Roman.woff') format('woff'), url('Delicious-
Roman.ttf') format('truetype'), url('Delicious-Roman.svg') format('svg');
```

Los formatos de fuente que pueden utilizarse con esta característica son:

- Truetype,
- OpenType
- Embedded OpenType
- WebOpen
- FontFormat (WOFF)

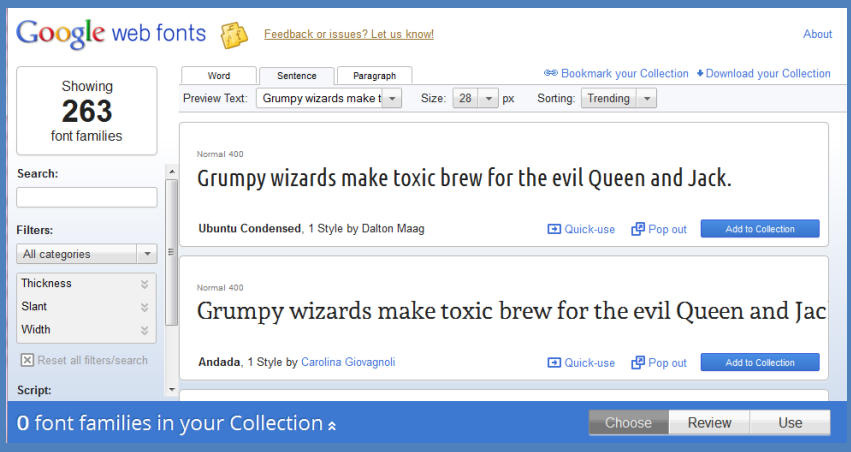
Por sus características de compresión y por estar pensaba especialmente para la web, se espera que WOFF se convierta en el estándar de tipografías para internet. Este formato se encuentra apoyado por el W3C (www.w3.org/TR/WOOF).

NOTA: un buen sitio en donde se pueden bajar Fuentes de uso gratuito en todos los formatos es: **http://www.fontsquirrel.com**



Otra opción es trabajar con el **API de Google**, esto nos evitará cargar la tipografía en nuestro servidor y nos dará la certeza que la tipografía elegida está en todos los formatos necesarios para lograr la mayor compatibilidad posible.

Podemos encontrarlo en **WWW.GOOGLE.COM/WEBFONTS**



PROPIEDAD WORD-WRAP

La propiedad ***word-wrap*** nos sirve para romper las palabras que son demasiado largas y no caben enteras por la anchura de una caja.

En el modelo de caja planteado desde html para los navegadores, las palabras se van ubicando en líneas para para cubrir el ancho disponible de la caja. Cuando tenemos palabras muy largas que no caben en nuestro contenedor, esto puede ser un problema. Dependiendo las propiedades de la caja contenedora del texto, la palabra saldrá fuera de la caja, o la caja se adaptará a la palabra, haciendo que se amplíe el ancho de la caja, esto en algunos casos puede alterar el formato de nuestro diseño (si teníamos div con posicionamiento flotante, probablemente caigan hacia abajo al redimensionar la caja de texto). En cualquier caso, el efecto resultante suele ser poco agradable, porque muchas veces nos descuadra nuestro diseño y hace que las páginas queden mal maquetadas.

Para evitar este efecto, en CSS 3 se ha incluido un atributo llamado ***word-wrap*** que sirve para especificar que las palabras que sean demasiado largas se deben cortar, de manera que quepan en el ancho disponible de la caja. Una propiedad muy útil cuando tenemos grandes bloques de texto dentro de nuestro contenido.

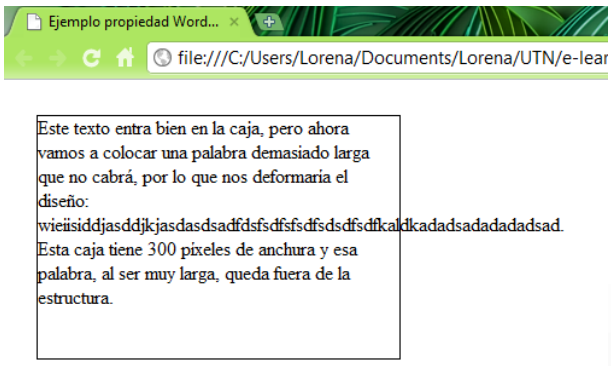
El atributo word-wrap tiene dos posibles valores: ***normal*** o ***break-word***.

- **word-wrap: normal;**

Hace que las palabras no se corten, lo que sería el comportamiento normal que conocíamos hasta ahora, ocurriendo que las palabras largas nos puedan descuadrar nuestro diseño.

Ahora podemos ver una caja que tenía una anchura de 300 px y que por culpa de una palabra muy larga se deforma la caja o el texto aparece por fuera.

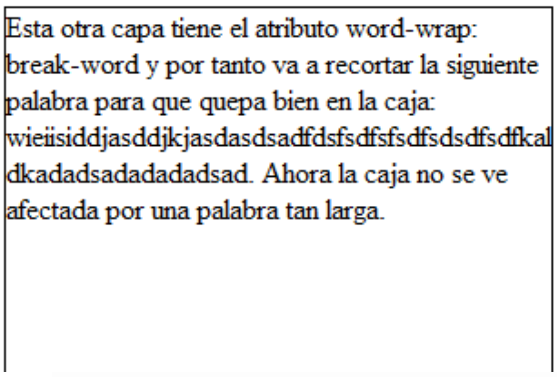
- **word-wrap: break-word;**



Vista previa en Google Chrome

Con este otro valor de ***word-wrap: break-word***, lo que conseguimos es que la palabra se recorte, para caber en el ancho que habíamos definido.

Este atributo por ahora es soportado por la mayoría de los navegadores (Internet Explorer, Safari, Firefox, Opera y Google Chrome), por lo tanto es una propiedad que ya puede comenzar a implementarse.



Vista previa en Firefox

TEXTOS MULTICOLUMNA

Numerosas publicaciones maquetan el texto en diversas columnas, como criterio de diseño y para facilitar la lectura. Podríamos imaginarnos el texto de los periódicos si no se encontrase dividido en diversas columnas ¿no sería casi imposible seguir las líneas para hacer una lectura cómoda de la información si todo estuviera en una única columna?

Este problema no lo encontramos generalmente en las páginas web, porque el texto que cabe en el cuerpo de la página no es tan grande como el que cabría en una hoja de un diario. Además, generalmente la propia página ya se encuentra dividida en diversas columnas. No obstante, aunque quisiéramos, con las características de CSS 2 no sería muy fácil hacer un texto fluído que se adaptase automáticamente en columnas, de modo que éstas crecieran homogéneamente a medida que el texto crece o se reduce. CSS 3 soluciona este problema, o más bien ofrecerá una solución al mismo, sencilla y automática.

Para crear una ***estructura multi-columna*** utilizaremos varios atributos, que servirán para modelizar las columnas:

colum-count: permite indicar en cuantas columnas se dividirá el contenido de un elemento contenedor.

colum-width: servirá para definir la anchura de las distintas columnas a crear.

column-gap: nos permitirá definir el espacio en blanco entre columnas.

column-rule: servirá para crear un filete o línea divisoria entre las columnas.

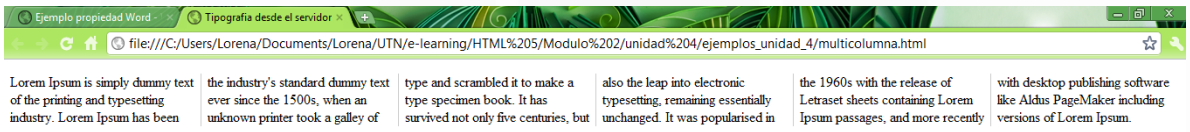
Estas etiquetas por el momento son soportadas solo por Opera. Sin embargo Safari, Google Chrome y Firefox ya implementan el multicolumna, de manera experimental y hasta que las especificaciones de CSS 3 estén dispuestas para incorporar en los navegadores. Para ello, podemos utilizarlas si colocamos los prefijos para cada navegador ("-moz-" en el caso de Firefox y "-webkit-" para el navegador Safari o Chrome.)

Un ejemplo de multicolumna, para que funcione en estos navegadores sería:

```
.multicolumna{
   -moz-column-width: 200px;
```

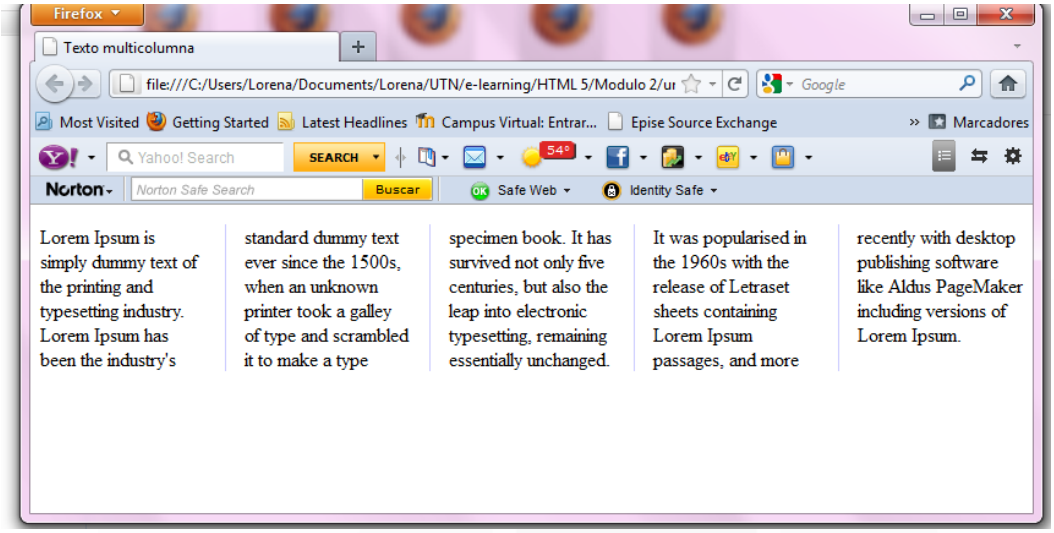
```
    -moz-column-gap: 15px;
    -moz-column-rule: 1px solid #ccc;
    -webkit-column-width: 200px;
    -webkit-column-gap: 15px;
    -webkit-column-rule: 1px solid #ccc;
    column-width: 200px;
    column-gap: 15px;
    column-rule: 1px solid #ccc;

  }
```

Visualización del ejemplo (multicolumna.html) desde Google Chrome:



Otra posibilidad para crear un multicolumna será definir simplemente el número de columnas que querríamos incorporar en el texto, por medio del atributo column-count, de esta manera:

```
  .multicolumna5columnas{
    -moz-column-count: 5;
    -moz-column-gap: 2em;
    -moz-column-rule: 1px solid #ccf;
    -webkit-column-count: 5;
    -webkit-column-gap: 2em;
    -webkit-column-rule: 1px solid #ccf;
    column-count: 5;
    column-gap: 2em;
    column-rule: 1px solid #ccf;
  }
```



Especificar el número de columnas es quizás más cómodo, pero en páginas fluidas puede funcionar peor, porque no se sepa con certeza qué anchura va a tener el lugar donde se muestran los textos y por tanto unas veces queden columnas muy anchas y otras muy estrechas.

El sistema de múltiples columnas se está encuentra en estado de borrador y forma parte de un módulo aparte dentro de las especificaciones de CSS 3. Podemos encontrar más información sobre el tema en la W3C desde la URL: http://www.w3.org/TR/css3-multicol/

TEXTOS CON SOMBRA

El atributo ***text-shadow*** se incluyó inicialmente en el CSS2 pero fué eliminado en CSS2.1. Renovado y con mayor compatibilidad con los navegadores, CSS3 vuelve a incluirlo. Este atributo es similar al atributo box-shadow que vimos en la unidad anterior. La diferencia es que, mientras que box-shadow crea sombras a en forma de caja, text-shadow realiza una sombra ajustada a los propios caracteres de un texto.

La sintaxis de *text-shadow* es igual a la de *box-shadow*:

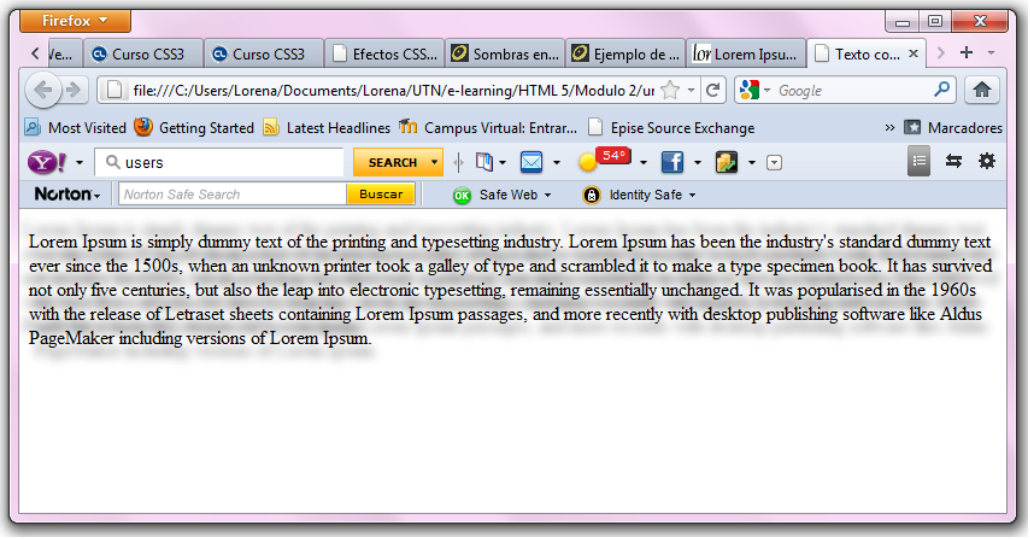
- **text-shadow: distanciaX distanciaY difuminado color;**

Por ejemplo:

```
text-shadow: 5px 10px 7px rgba(0,0,0,0.5);
```

Para asignar varias sombras usamos la coma como separador. Por ejemplo:

```
text-shadow: 5px 10px 7px rgba(0,0,0,0.5), -5px -10px 7px rgba(0,0,0,0.5);
```



TEXTOS EN 3D CON TEXT-SHADOW

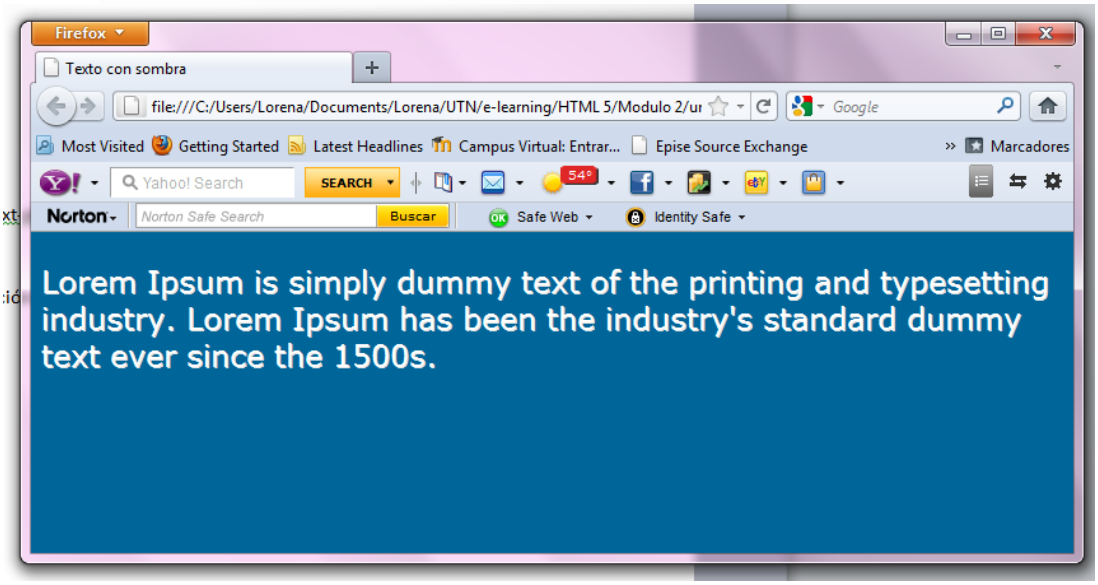
Veamos una aplicación práctica de *text-shadow* que dará como resultado un efecto distinto a una simple sombra paralela.

Sabemos que podemos aplicar diferentes sombras a un texto, y que podemos decidir que el difuminado de la sombra sea 0. Partiendo de esta base, podemos crear un efecto de pseudo 3D a cualquier texto.

Primero tenemos que aplicar el efecto sombra a nuestro texto. Creamos un texto blanco y una sombra a una distancia *x* e *y* de un color gris claro y con difuminado 0:

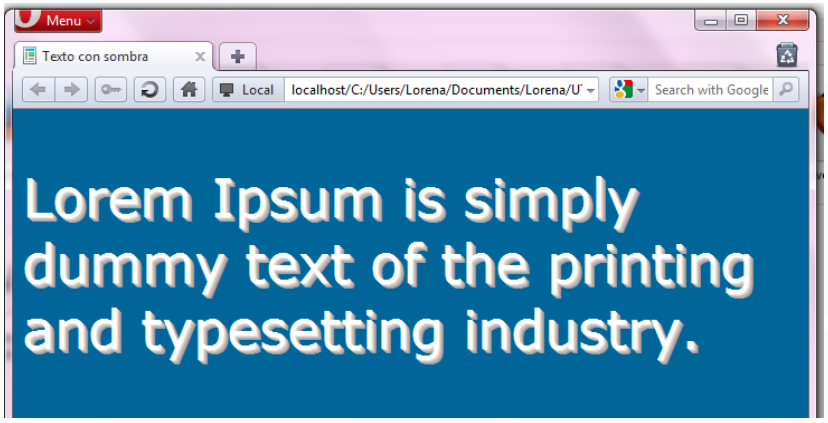
Por ejemplo:

```
text-shadow: 1px 1px 0px rgba(230,230,230,1);
```



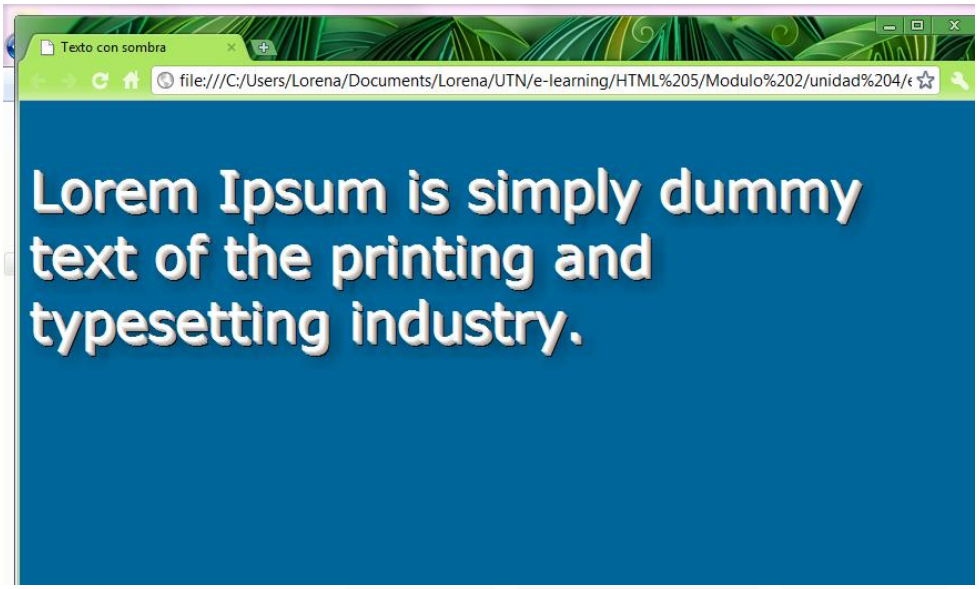
Podemos intensificar el efecto, agregando algunas sombras más:

```
text-shadow:
1px 1px 0px rgba(230,230,230,1),
2px 2px 0px rgba(200,200,200,1),
3px 3px 0px rgba(180,180,180,1),
4px 4px 0px rgba(160,160,160,1);
```



Si queremos hacerlo aún más realista, podemos aplicar una nueva sombra siguiendo el método anterior, con la diferencia que esta vez le daremos color negro. Por último añadimos una sombra mayor que todas las demás, de color gris claro y con difuminado:

```
text-shadow:
1px 1px 0px rgba(230,230,230,1),
2px 2px 0px rgba(200,200,200,1),
3px 3px 0px rgba(180,180,180,1),
4px 4px 0px rgba(160,160,160,1),
/*Añadimos*/
5px 5px 0px rgba(0,0,0,1),
8px 8px 20px rgba(0,0,0,0.5);
```


Texto con sombra
file:///C:/Users/Lorena/Documents/Lorena/UTN/e-learning/HTML%205/Modulo%202/unidad%204/
Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Resumen

En esta Unidad…

En la presente unidad desarrollamos los conceptos necesarios para incorporar los atributos gráficos a nuestras estructuras de HTML utilizando el lenguaje CSS3

Con las propiedades propuestas podemos utilizar cualquier familia tipográfica para nuestra web. También trabajamos con la posibilidad de desarrollar sitios adaptables a los diferentes dispositivos disponibles hoy en el mercado.

En la próxima Unidad…

En la próxima unidad vamos comenzar a trabajar con los conceptos de programación, para prepararnos para incorporar contenido dinámico a nuestros sitios.