

# **MÓDULO**

## ***Diseño de Interfaces Web***

### **UNIDAD 2-4**

Responsive Design con mediaquerys

## 1. Introducción

El diseño web adaptable o adaptativo (en inglés, Responsive Web Design) es una filosofía de diseño y desarrollo web que, mediante el uso de estructuras e imágenes fluidas, así como de @media queries en la hoja de estilo CSS, consigue adaptar el sitio web al entorno del usuario.



## 2. Ventajas

La principal ventaja que encontramos con el diseño web adaptable es que la web se visualizará correctamente en todos los dispositivos que usemos y se adaptará a los giros en dispositivos móviles.

Además, Google tiene en cuenta las páginas web que tienen diseños sensibles o adaptables, gracias a su Googlebot-Mobile.

Se recomienda el uso de media queries para crear las páginas web adaptativas.

La alternativa sería usar distinto código según el dispositivo, o incluso, distintas URLs.

Con las media queries Google no necesita analizar tanto contenido de la misma web, lo que le facilita la labor de asignar el tipo de contenido de la página y de esta forma mejorará el SEO de la web.

### 3. Viewport

Esta meta-etiqueta fue creada en principio por Apple para su iPhone, pero se ha convertido en un estándar que es soportado por la mayoría de los dispositivos móviles.

Su uso es totalmente necesario, ya que, sino el navegador establece el ancho con el que prefiere visualizar una página, en lugar de usar el ancho del que dispone, es decir, si la pantalla de nuestro móvil tiene 400px y el navegador detecta que lo óptimo sería visualizarla con 700px, así lo hará, a no ser que usemos este meta).

Como siempre, el meta lo añadiremos en el <head> de nuestra página:

```
<meta name="viewport" content="width=device-width,  
initial-scale=1.0, user-scalable=no">
```

### 3. Viewport

Se pueden usar los siguientes parámetros (separados por comas):

- width: ancho de la página (se puede establecer en píxeles o como device-width y usará el ancho del que dispone el dispositivo).
- height: alto de la página, actúa igual que el width.
- initial-scale: escala o zoom inicial de la página (este y los demás tipos de escala se establecen con valores como 1.0 para no tener zoom o 2.5 para tener un zoom de 2 aumentos y medio, por ejemplo).
- minimum-scale: zoom mínimo que podemos hacer en la página.
- maximum-scale: zoom máximo que podemos hacer en la página.
- user-scalable: establece si está permitido o no hacer zoom (yes/no).

## 5. Preparar nuestra página para el diseño adaptable

Para lograr que nuestra web se adapte a los anchos de pantalla, debemos tener en cuenta varias cosas importantes:

1. La primera, y la más importante, es dejar de usar píxeles en todos los sitios, en su lugar, usaremos porcentajes (por ejemplo: width: 60%). Supongamos que tenemos un elemento con un ancho de 960 píxeles y, dentro de este, tenemos otro elemento de 300 píxeles. Al elemento externo le pondremos, por ejemplo, un width del 90% para que se adapte al tamaño de la ventana y para que el elemento interno mantenga la proporción con respecto al externo calcularemos el ancho que debería tener así:  $300/960$ , lo cual, nos dará 0,3125. Por lo tanto, el width que le pondremos al elemento interno será 31,25%.
2. Para limitar el ancho (o alto si se terciara) debemos de usar el parámetro max-width (max-height en el caso del alto). Para establecer el mínimo usaremos min-width y min-height, aunque estos se utilizan menos. El valor

## 5. Preparar nuestra página para el diseño adaptable

de estas propiedades sobrescribirá el valor de las propiedades width y height. Por ejemplo, si indicamos width: 100% y max-width: 600px, cuando visualicemos la página con una resolución de 1200px, el ancho será de 600px, ya que es el valor máximo que hemos indicado para el ancho En la siguiente imagen podemos observar cómo se muestra a continuación:

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

width = 80% - max-width = 600px

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

width = 100% - max-width = 320px

## 5. Preparar nuestra página para el diseño adaptable

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

`width = 100% - max-width = 50%`

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

`width = 100% - max-width = 40em`

2. Nunca debemos permitir que una imagen de fondo que está pensada para no repetirse, llegue a repetirse por el cambio de dimensiones. Para evitarlo, debemos adaptarla, normalmente, usando `@media-queries`.
3. Tampoco debemos permitir que las imágenes y los vídeos se salgan de la estructura, sino aparecerá un scroll lateral en los dispositivos móviles que destruirá totalmente el diseño.



## 6. @media queries

Desde CSS 2.1, nuestras hojas de estilos usaban los media types para distinguir entre diferentes dispositivos. Por ejemplo, cuando creamos una hoja de estilos para impresión:

```
<link rel="stylesheet" type="text/css" href="core.css"
media="screen" />
<link rel="stylesheet" type="text/css" href="print.css"
media="print" />
```

Pero en muchos navegadores no funcionan todos media types de forma correcta.

El W3C creó las media queries como parte de la especificación CSS3.

**Una media query nos permite inspeccionar las características físicas del dispositivo que está renderizando nuestro trabajo.**

## 6. @media queries

Para hacerlo, podemos incorporar una query al atributo media de una hoja de estilos linkeada:

```
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 480px)" href="shetland.css" />
```

La query contiene dos componentes: Un media type (screen), y la consulta entre paréntesis, que contiene una característica a inspeccionar (max-device-width) seguida por el valor al que apuntamos (480px). En otras palabras, le estamos preguntando al dispositivo, si su resolución horizontal (max-device-width) es igual o menor que 480px.

En el caso de que visualicemos la página en un dispositivo con una pantalla pequeña cargará shetland.css. De lo contrario, el link se ignora y no lo carga.

## 6. @media queries

Podemos testear múltiples valores de las propiedades en una sola query, encadenándolos con la palabra clave `and`, por ejemplo:

```
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 480px) and (resolution: 163dpi)" href="shetland.css"/>
```

Además del uso de la etiqueta `<link>` para incluir media queries, también tenemos la posibilidad de incluirlas en nuestro CSS como parte de una regla `@media`:

```
@media all and (max-device-width: 480px) {  
    .column { float: none; }  
}
```

O como parte de una directiva `@import`:

```
@import url("shetland.css") all and (max-device-width: 480px);
```

La palabra `all` indica que la regla será para todos los tipos de medios (*ver ejemplo01-mediaqueries y ejemplo02-mediaqueries*)

## 6. @media queries

Aunque las @media queries nos permite conocer muchas propiedades de los dispositivos, las más importantes son las siguientes:

- aspect-ratio: Hace referencia a las dimensiones del dispositivo expresadas como una relación de aspecto: 16:9 por ejemplo.
- width y height: Se refiere a las dimensiones del área de visualización. Además pueden ser expresadas en valores mínimos y máximos.
- orientation: Detecta si es panorámico (el ancho es mayor que el alto) o vertical (el alto es mayor que el ancho). Esto nos permite ajustar los diseños para dispositivos con propiedades de giro de la pantalla como smartphones o los tablets.

## 6. @media queries

- resolution: Indica la densidad de los pixeles en el dispositivo de salida. Es útil para los dispositivos que tiene una resolución mayor a 72 dpi.
- color, color-index y monochrome: Detectan el número de colores o bits por color. Esto nos permite crear diseños específicos para dispositivos monocromáticos.

## 7. Puesta en marcha del Responsive Design

En primer lugar, nos olvidaremos de los float y usaremos en su lugar la propiedad display (a ser posible flexbox).

Una de las cosas que se dan mucho en los diseños adaptativos es crear un conjunto de bloques con imágenes y texto que, según el tamaño de la ventana, se van recolocando.

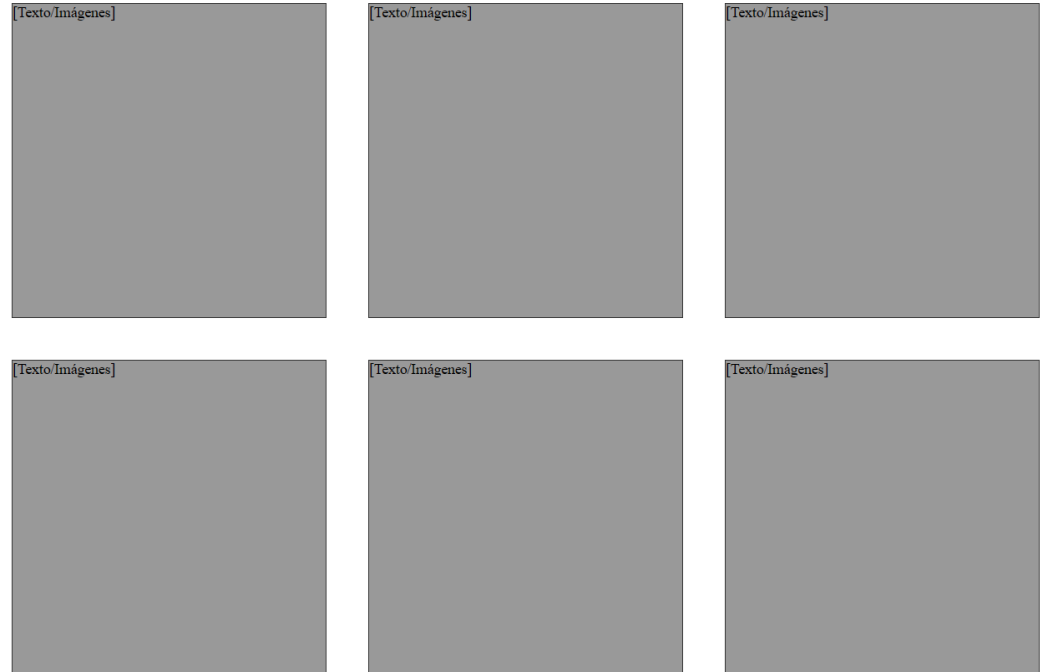
Por ejemplo, si tenemos seis bloques, tres encima y tres debajo, pero si los visualizáramos con un ancho de pantalla mayor queremos que se vean en línea. Podríamos tener en el html:

```
<div id="contenedor">
  <figure class="bloque">[Texto/Imágenes]</figure>
  <figure class="bloque">[Texto/Imágenes]</figure>
  <figure class="bloque">[Texto/Imágenes]</figure>
  <figure class="bloque">[Texto/Imágenes]</figure>
  <figure class="bloque">[Texto/Imágenes]</figure>
  <figure class="bloque">[Texto/Imágenes]</figure>
</div>
```

## 7. Puesta en marcha del Responsive Design

En la hoja de estilos, podríamos especificar, por ejemplo:

```
#contenedor {  
  display: flex;  
  flex-wrap: wrap;  
}  
#contenedor .bloque {  
  height: 300px;  
  width: 300px;  
  border: 1px solid #333;  
  background: #999;  
  margin: 20px;  
}
```



Si probamos a ampliar y reducir el tamaño de la ventana, veremos que los elementos se van reubicando automáticamente.

## 7. Puesta en marcha del Responsive Design

¿Qué ocurre si queremos que los elementos se comporten de una forma diferente cuando el tamaño de ventana disponible sea inferior a un valor determinado? (ver ejemplo03-mediaqueries)

En ese caso, utilizaremos las media queries. Por ejemplo, si quisiéramos que a partir de los 800 píxeles los bloques anteriores se mostraran unos debajo de otros, centrados y ajustando su ancho al espacio disponible, haremos lo siguiente:

```
@media screen and (max-width: 800px) {  
    #contenedor {  
        flex-direction: column;  
    }  
    #contenedor .bloque {  
        width: auto; /*Reescribimos*/  
    }  
} /*Si hace falta usamos !important para que tenga preferencia*/
```



## 8. Pasar de columnas a cascada

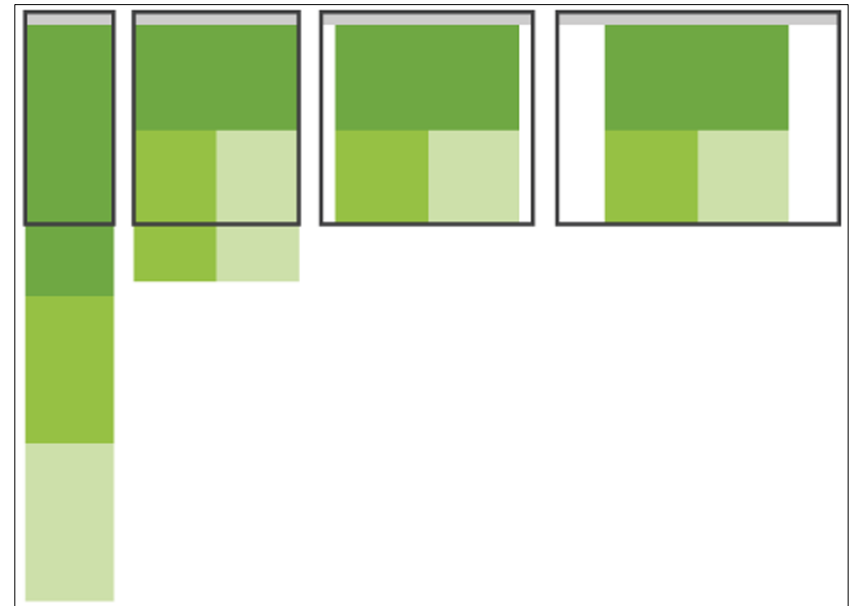
Uno de los métodos que se utiliza más a menudo para cambiar la estructura de una web adaptativa, que suele tener una cabecera y dos o tres columnas (con el menú, el contenido, enlaces, etc.) es el paso de la disposición en columnas a la disposición en cascada.

En la siguiente url

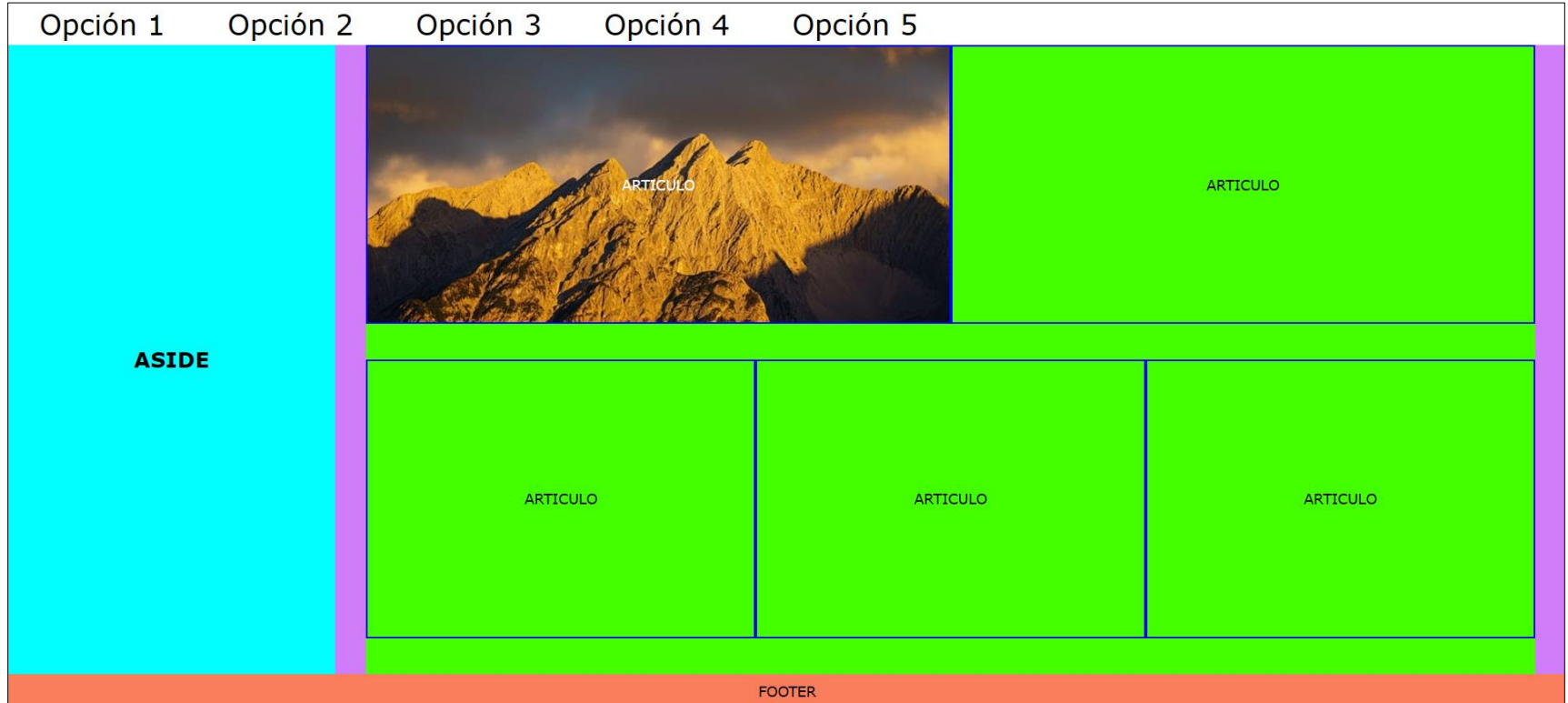
(<http://www.hsgac.senate.gov/>)

podemos ver un ejemplo que utiliza esta técnica.

En las siguientes imágenes se puede ver cómo, en función del ancho disponible se cambia la disposición.



## 8. Pasar de columnas a cascada



## 8. Pasar de columnas a cascada

Opción 1

Opción 2

Opción 3

Opción 4

Opción 5

**ASIDE**



ARTICULO

ARTICULO

ARTICULO

FOOTER

## 8. Pasar de columnas a cascada

Veamos cómo hacer una estructura adaptable similar a la de las imágenes anteriores.

En primer lugar, es habitual adoptar un procedimiento “mobile first” en el que primero programamos el tamaño para móvil.

Una vez programado el tamaño de móvil, se añaden las media queries para tamaños de pantalla mayores a móvil y dentro de las media queries se reescribe todo lo necesario para el tamaño de Tablet.

Después se añaden otras mediaqueries para tamaños mayores de tablet y así sucesivamente con nuevas media queries según los “puntos de ruptura” o tamaños de dispositivos que queramos programar. (*ver ejemplo04-mediaqueries*)

## 9. Adaptar imágenes y vídeos

Muchas veces, para crear diseños muy optimizados puede ser importante usar varias imágenes distintas, una con un tamaño más pequeño y otra con uno superior. Esto lo podemos hacer, como en el ejemplo visto, cambiando la propiedad background-image.

Si no modificamos la imagen usaremos la etiqueta img y podemos hacer que una imagen se comporte de las siguientes formas:

- Ocupar el ancho de la página. Puede ser útil para cabeceras o imágenes principales y lo haremos estableciendo su ancho al 100%.

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

- Tener una imagen con un tamaño máximo que no sobrepasará, pero cuando se disminuya el tamaño del contenedor se encogerá adaptándose a la página.

```
img { width:100%; max-width:400px; }
```

## 9. Adaptar imágenes y vídeos

- Tener las imágenes con su tamaño original como máximo. Este caso es muy similar al anterior, con la diferencia que pondremos como tamaño máximo el ancho de la imagen. De esta forma, la imagen no se deformará al ser redimensionada a un tamaño superior al original.

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

Para adaptar el tamaño de los vídeos usaremos las mismas técnicas que utilizamos para las imágenes.

Si hemos puesto un alto fijo a un vídeo, y después, en una @media query, ponemos, por ejemplo, un ancho relativo utilizando porcentajes, el elemento <video> se redimensionará para mantener las proporciones al redimensionar la ventana, pero el espacio reservado para el alto inicial se mantendrá reservado, por lo que los

## 9. Adaptar imágenes y vídeos

elementos que se encuentren alrededor no se reubicarán. Para solucionar este problema, bastará con indicar en la @media query que el alto será automático (height:auto).

## 10. La barra de navegación

En las versiones para menores resoluciones o tamaños de pantalla el menú suele convertirse hoy en día, casi un estándar de facto, en un icono con tres rayitas que se despliega.

Pasamos de tener un menú lineal y visible a un menú desplegable. A continuación, veremos cómo podemos implementar este menú basándonos en un ejemplo. Tenemos el siguiente código html:

```
<nav>
  <a href="#">Menú</a>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Collection</a></li>
    <li><a href="#">Blog</a></li>
    <li><a href="#">Contact</a></li>
    <li><a href="#">Directions</a></li>
  </ul>
</nav> <script src="../js/menu.js"></script>
```



## 10. La barra de navegación

A este código le aplicaremos los siguientes estilos:

```
* {
    margin: 0;
    padding: 0;
}
nav {
    width: 100%;
    background-color: #fff;
    border-bottom: 1px solid #ccc;
}
nav ul {
    list-style: none;
    padding: 0px;
    margin: 0px;
    font-weight: bold;
    text-align: center;
}
nav ul li {
    display: inline-block;
    text-align: left;
}
nav ul li a {
    display: block;
    padding: 15px 10px;
    text-decoration: none;
    color: #444;
}
nav ul li a:hover {
    background-color: #ccc;
}
nav>a {
    display: none;
}
```

## 10. La barra de navegación

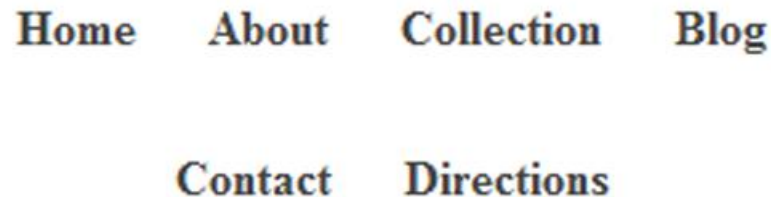
Con estos estilos nuestra barra de navegación quedará como se muestra en la siguiente imagen:



Home About Collection Blog Contact Directions

Se puede observar que el enlace con el texto Menú queda oculto (display:none).

Si reducimos la ventana del navegador, los enlaces de la barra se reubicarán de la siguiente forma:



Home About Collection Blog

Contact Directions

## 10. La barra de navegación

Lo cual, podría no ser deseable. Para mejorarlo, vamos a hacer que cuando el tamaño de la ventana sea menor de 475 píxeles, el menú se oculte y se muestre sólo el enlace con el texto Menú. Después, al hacer click con el ratón sobre este enlace el menú se desplegará y al volver a pulsar se plegará.

Añadiremos la siguiente @media query:

```
@media all and (max-width: 475px) {  
  nav ul {  
    display: none;  
  } /* ocultamos el menú de navegación */  
  nav > a /* mostramos el enlace con el texto Menú */ {  
    display: block;  
    padding: 0 1em 0;  
    text-align: center;  
    padding: 10px 15px;  
  }  
}
```

## 10. La barra de navegación

```
    color: #fff;
    background-color: #0084b4;
    text-decoration: none;
    margin: 3px;
}
/* Con la clase desplegado el menú se mostrará verticalmente */
ul.desplegado {
    display: block;
    list-style: none;
}
ul.desplegado li {
    display: block;
    text-align: center;
}
ul.desplegado li a {
    display: block;
    border-bottom: 1px solid #ccc;
}
}
```

## 10. La barra de navegación

Con esta @media query cuando reducimos la ventana el menú se mostrará así:



Ya sólo nos queda hacer que al pulsar el enlace el menú se despliegue. Para ello, utilizaremos el siguiente código javascript:

```
var enlaceMenu;
function iniciarMenu() {
    enlaceMenu = document.querySelector("nav>a");
    enlaceMenu.addEventListener("click", despliegaMenu, false);
}
function despliegaMenu() {
    document.querySelector("nav>ul").classList.toggle("desplegado");
}
window.addEventListener("load", iniciarMenu, false);
```

## 10. La barra de navegación

En el código anterior, al pulsar el enlace menú se le aplicará la clase desplegado al <ul> si no la tiene aplicada o se eliminará dicha clase si ya la tiene aplicada. Esto lo hacemos con el método `classList.toggle('desplegado')`. Con esto, al pulsar sobre el enlace se desplegará el menú y quedará así:  
(ver *ejemplo05-mediaqueries*)

Menú
Home
About
Collection
Blog
Contact
Directions