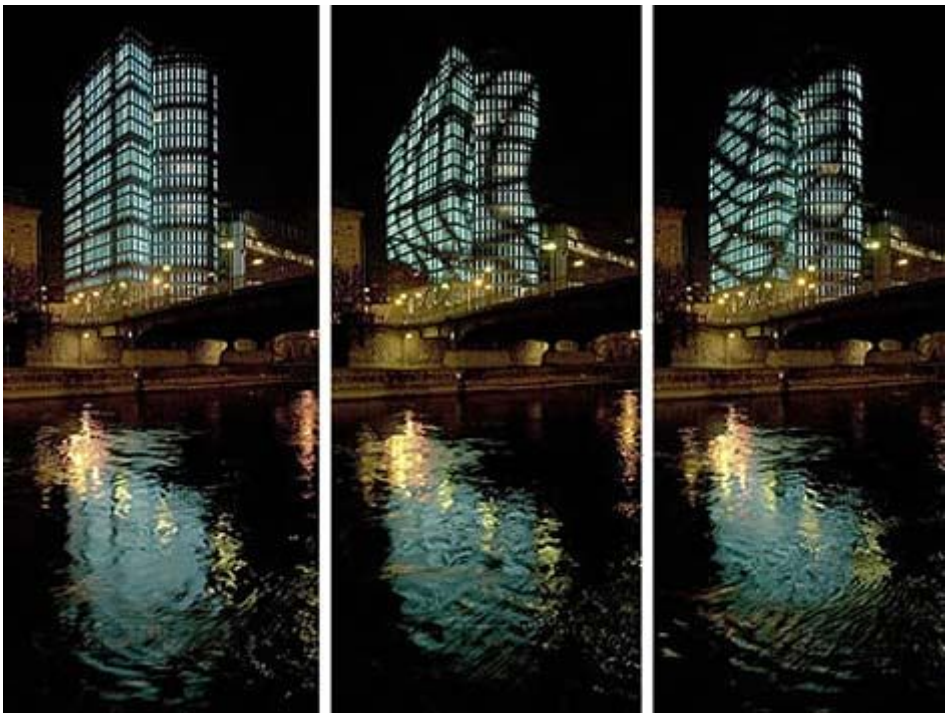
DAWEB
Práctica 8, día 12 de Abril 2018

Páginas web con diseño adaptable

1. Introducción

Tomemos dos ejemplos simples. El primero podrían ser las ventanas que se vuelven opacas según la luminosidad externa. O esas paredes de espejo donde cuanto más nos acercamos, más parecen deformarse. En resumen, lo que los arquitectos buscan en este movimiento es ir más allá/abstraerse de las limitaciones inherentes a los diferentes soportes. En el mundo de la arquitectura, podemos afirmar que estos soportes son realmente múltiples y numerosos.



Mediante un simple juego de luces, este edificio parece tener una estructura diferente. Fuente: http://www.essential-architecture.com/STYLE/STY-075.htm

El diseño adaptable o *Responsive Web Design* nació porque cada vez nos enfrentamos más a los mismos problemas que los arquitectos: adaptarse a los medios.

Considera que en el pasado, solo teníamos uno o dos navegadores y los tamaños de pantalla variaban poco. Ahora hay multitud de navegadores y un gran número de resoluciones (desde 240x320 hasta 2515x1886 o más). Pero también los medios de

visualización: podemos acceder a internet desde nuestros clásicos ordenadores de sobremesa, pero también a través de smartphones, tabletas, relojes etc. Sin embargo, sabemos que la manera de mostrar un sitio web difiere de la de un ordenador de escritorio.

La web actual. Fuente: http://bradfrostweb.com/blog/post/this-is-the-web/

Y todo esto es lo que tenemos ahora. Poco a poco estamos empezando a acceder a internet a través de televisores (de forma directa o a través de consolas o adaptadores específicos). Empiezan a aparecer prototipos, con internet en las radios de los coches, en los relojes, en los coches, en los frigoríficos... En definitiva, un nuevo mundo al que todavía tendremos que adaptarnos.

La web del futuro. Fuente: http://bradfrostweb.com/blog/post/this-is-the-web/

2. Las tres nociones del "*Responsive Web Design*"

Para hacer una aplicación web que sea adaptable, debemos establecer los siguientes puntos:

- una cuadrícula de visualización flexible: es decir, una plantilla que no dependa de una resolución mínima y máxima;
- soportes flexibles: es decir, asegurarse que las imágenes, vídeos (si es necesario), no van más allá del marco de nuestra cuadrícula de visualización/plantilla;
- un conjunto de reglas CSS basadas en *Media Queries* (Media Query es una característica de CSS que permite al contenido de un sitio web adaptarse a diferentes tamaños de pantalla y resoluciones): cuyo principio es poner condiciones en la pantalla para mostrar/ocultar o incluso cambiar el renderizado de la aplicación web.

Si hacemos una analogía con la arquitectura, los tres puntos representan nuestras herramientas. En lo que se refiere a nuestros materiales, esto corresponderá a

nuestros navegadores. Así como algunos materiales son mejores que otros. Por ejemplo, la madera de mala muerte de IE6 es menos robusta que el hormigón reforzado de Chrome. Nuestras herramientas, por correctas que sean, no siempre serán fáciles de aplicar. Porque existe una idea falsa y se resiste: no es posible hacer diseño adaptable en navegadores antiguos.

Afortunadamente, es posible adaptarse para conseguirlo. Sin embargo, el coste será (inevitablemente) mayor y será necesario el uso de JavaScript.

3. Rejilla de visualización flexible

En primer lugar, es necesario crear un sitio cuya cuadrícula de visualización sea flexible. En otras palabras, nunca (o al menos tan poco como sea posible) se usará un ancho/alto fijo.

Para empezar, vamos a dimensionar el sitio en la llamada resolución "óptima", es decir, la resolución para la que la pantalla es más adecuada (por ejemplo, 1280 * 800 píxeles).

Una vez hecho esto, cambiaremos tanto el tamaño de la fuente como el tamaño de nuestros elementos. Nos aseguraremos de que todo se exprese como un porcentaje (en el caso de las fuentes, usaremos lo que llamamos "em", donde 1em = 100%).

Como regla general, estimamos que el tamaño por defecto de nuestro texto (para que sea legible) es de 16 píxeles. Suponemos que 16 píxeles corresponden a un 1em. Si, en nuestra hoja de estilo, tenemos un título cuyo tamaño de fuente es de 24 píxeles, reemplazaremos este valor por 1.5em. De hecho, debemos aplicar la siguiente fórmula cada vez:

```
objetivo / contexto = resultado
```

El objetivo corresponde al tamaño deseado, el contexto al tamaño habitual y obviamente, el resultado se expresa en "em". Es interesante que si el contexto cambia (por resolución), el texto permanece proporcional a la pantalla.

Una buena práctica es poner al lado del texto la fórmula para recordar por qué el resultado:

```
h1 {
font-size: 1.5em; /* 24 / 16 pixels */
}
```

Sin embargo, no siempre es fácil convertir tamaños de fuente ya que podemos manipular puntos, píxeles, em o porcentajes. El sitio http://pxtoem.com/ ofrece una tabla básica, así como un convertidor.

Antes de ver nuestros elementos, es de destacar un punto importante: a veces, tendremos resultados sorprendentes, del estilo: 0.33333333333333333333... En resumen, tropezar con un número infinito. Tendremos tendencia a redondear esto, a 0,34 por ejemplo, eso, en sí mismo, es un error. Es aconsejable colocar tantos números como sea posible después del punto decimal (obviamente no cientos, pero una docena o quince no es excesivo). Los navegadores son lo suficientemente inteligentes como para redondear. Y así, tendremos una pantalla optimizada.

Ahora, para nuestros elementos, aplicaremos exactamente el mismo principio. Imaginemos el siguiente caso: tenemos nuestro sitio web que tiene 1018 píxeles de ancho en el caso ideal. Luego tenemos un área principal de 766 píxeles y un menú de 252 píxeles. Entonces los declararemos de la siguiente manera:

```
.menu {
width: 24.7544204322%; /* 254 / 1018 pixels */
}

.main {
width: 75.2455795678%; /* 766 / 1018 pixels */
}
```

Sin embargo, tendremos una diferencia notable en comparación con la fuente: el contexto variará. De hecho, el tamaño de fuente habitual será el mismo en todas partes en nuestra página. No podemos decir lo mismo de nuestra anchura y altura. En otras palabras, debemos reajustar nuestro contexto de acuerdo a nuestros entornos.

Por ejemplo, si continuamos con nuestro caso, imaginemos que la zona principal en sí tiene dos subzonas. Su contexto irá de 1018 píxeles (ancho total de página) a... 766 píxeles.

```
.main-left {
width: 24.543080939%; /* 188 / 766 pixels */
}

.main-right {
width: 75.456919061%; /* 578 / 766 pixels */
}
```

Y tendremos que aplicar este principio en toda nuestra página (en la medida de lo posible), así como en las alturas, anchuras, márgenes (« margin », « padding ») y bordes.

Por supuesto, a una cierta resolución, o no habrá suficiente espacio para mostrar las imágenes, el texto... o habrá demasiado espacio y tendremos mucho espacio vacío que llenar. Los dos puntos siguientes están ahí para compensar esto.

4. Medios flexibles

Para asegurarnos de que nuestro sitio web puede ser adaptable, necesitamos asegurarnos de que los medios (imágenes, vídeos,...) también lo son. No hay nada más desagradable que ver una imagen más allá de su marco, haciendo poco estético nuestro sitio web.

Para resolver este problema, CSS nos ayudará, existe una propiedad llamada «max-width» que permite especificar el ancho máximo del elemento, con respecto a su padre.

Como resultado, no es raro que coloquemos las siguientes CSS en nuestros archivos:

```
img, object, embed, canvas, video, audio, picture {
max-width: 100%;
height: auto;
}
```

De esta manera, generalizamos el «max-width» en todos los medios de nuestro sitio web. Incluso si especificamos un ancho irrazonable en una imagen, siempre y cuando no cambiemos la propiedad «max-width», nunca irá más allá de su marco. Hay que tener en cuenta que el uso de «height : auto» nos permite mantener las proporciones correctas de nuestros soportes (de lo contrario se distorsionarían).

Sin embargo, hay una trampa. De hecho, incluso si esta propiedad es parte de la especificación CSS2 y existe desde hace mucho tiempo, IE6 (nuestro enemigo mortal) obviamente no lo reconoce, mientras que otras versiones de IE y otros navegadores lo reconocen bien.

Por lo tanto, para compensar esto, especificamos una herramienta de CSS que establecerá el ancho al 100%:

```
img, object, embed, canvas, video, audio, picture {
max-width: 100%;
height: auto;
width: 100%; /* IE6  */
}
```

Sin embargo, esta solución tiene sus inconvenientes: el material siempre ocupará todo el ancho y si especificamos uno, puede desbordarse. Aunque IE6 se considera muerto (no ha sido mantenido por Microsoft), todavía se utiliza en grandes cuentas y lo encontramos en las especificaciones de nuestros proyectos.

En este caso, podemos cargar una librería JavaScript que le obligará a reconocer esta propiedad. Este es particularmente el caso de este proyecto OpenSource: http://code.google.com/p/ie7-js/.

Contiene archivos JavaScript para IE8, IE9 y también para versiones entre IE5 e IE7. Tratan de corregir algunas deficiencias (como los problemas de transparencia en IE6 e IE7). También permite corregir el «max-width».

Así que, gracias a esto, tenemos nuestros medios (*media queries*) que ahora son adaptables y para una amplia gama de navegadores y versiones.

5. Medios (*media queries*)

Hemos visto cómo definir las plantillas de nuestros sitios y asegurarnos de que no se basan en métricas fijas, entonces nos hemos asegurado de que nuestros medios no perturben la visualización de nuestros elementos.

Ahora, nos aseguraremos de adaptar la pantalla de acuerdo a la resolución. De hecho, tenemos más opciones de colocación en una resolución de 1280x800 píxeles que en una resolución de 320x240 píxeles. Por lo tanto, debemos asegurarnos de que nuestros elementos entren en estas distintas resoluciones, aunque esto signifique no mostrar algunas de ellas, o incluso mostrarlas en otro lugar o mostrarlas de forma diferente.

Esto implica en alguna parte que tenemos que poner condiciones a los estilos que tenemos que aplicar en nuestros sitios web. Y eso es exactamente lo que vamos a usar, a través de las *media queries*.

Esta es una característica que nos trae CSS3. Permite (como su nombre indica) establecer una condición a:

- importar un fichero de CSS
  <link rel="stylesheet" media="screen and (color)" href="ejemplo.css" />
- un bloque CSS
  ```
  @media screen and (min-width:500px) {
     body {
        background-color: white;
     }
  }
  ```
- o también vía JavaScript
  ```
  if (window.matchMedia("(min-width: 400px)").matches) {
     // Se convierte el menú a colapsable}
  ```

Hay muchas posibilidades para establecer nuestras condiciones. Podemos hacerlo según el tipo de soporte (pantalla, impresora, tv...), pero también según las características del soporte, como la relación, orientación, colores....

Como regla general, utilizaremos las siguientes condiciones:

- orientación (para saber si estamos en modo vertical u horizontal);
- min-width, max-width (para determinar la anchura mínima/máxima);
- min-height, max-height (para determinar la altura mínima/máxima, sus usos son más raros, preferimos las dos propiedades anteriores).

Antes de ir más lejos, tenemos que decidir cómo mostrar nuestros elementos de acuerdo a la orientación, pero también de acuerdo a los diferentes tramos de resolución que manejaremos.

En el modo retrato, comúnmente estimamos que :

- un ancho de pantalla de menos de 480px viene de un *smartphone;*
- un ancho de pantalla entre 480px y 1024px viene de una tableta;
- un ancho de pantalla superior a 1024px proviene de ordenadores de sobremesa o portátiles.

Estas reglas no son fijas, obviamente hay teléfonos inteligentes, tabletas y ordenadores que las contradicen.

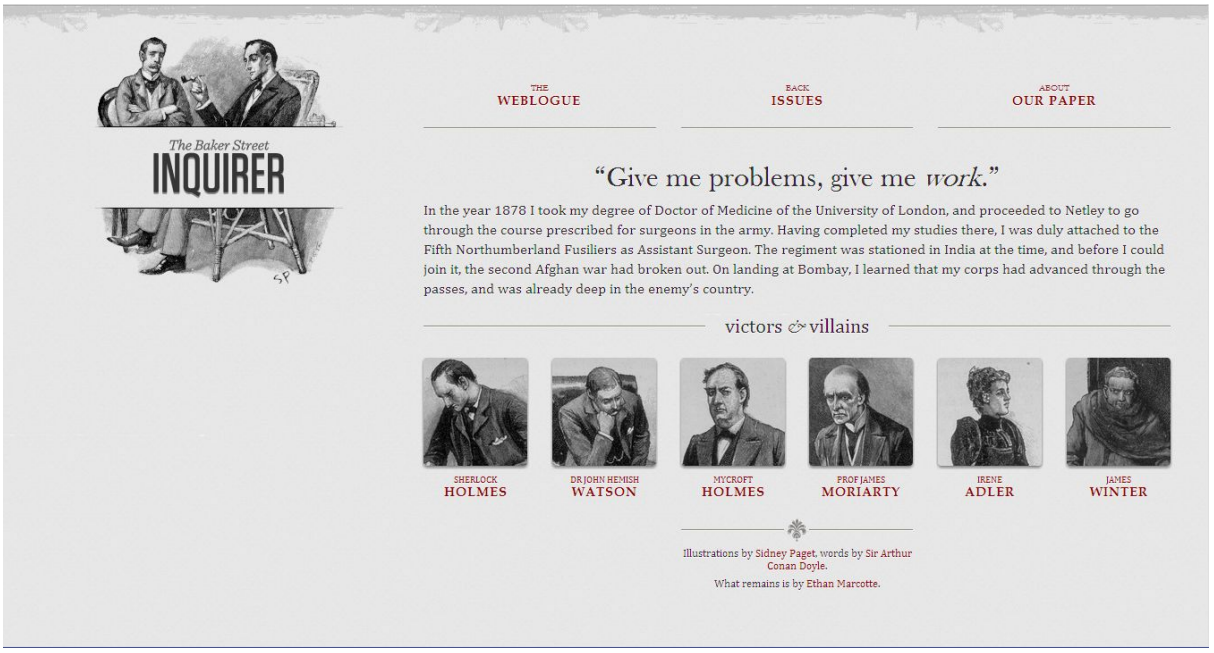
Una vez decididos los cortes de resolución, cómo queremos mostrar nuestros elementos en relación a ellos (y también en relación a la orientación), simplemente colocamos el código CSS en esos bloques para obtener el efecto deseado.

Algunos dirán: "Es CSS3, solo funciona en navegadores recientes". “Las *Medias Queries* están presentes... excepto en IE” (de hecho, empieza a funcionar más o menos en IE8).

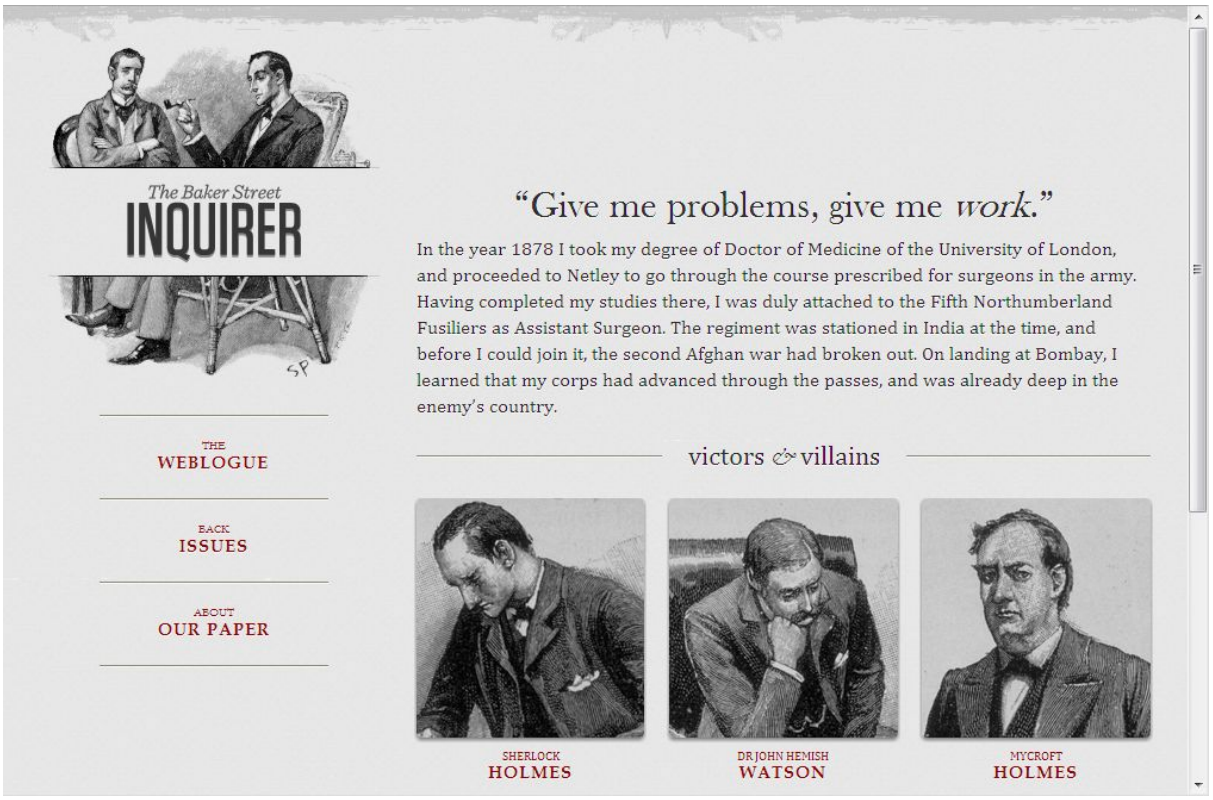
Afortunadamente para nosotros, existe un famoso polyfill HTML5 que permite ejecutar las *Media Queries* con estas dos propiedades en IE6 e IE7. Se llama Respond.js y podemos encontrarlo en el siguiente enlace de GitHub: https://github.com/scottjehl/Respond.

También hay «CSS3-mediaqueries» que hace que todas las *Media Queries* para IE funcionen: http://code.google.com/p/css3-mediaqueries-js/.

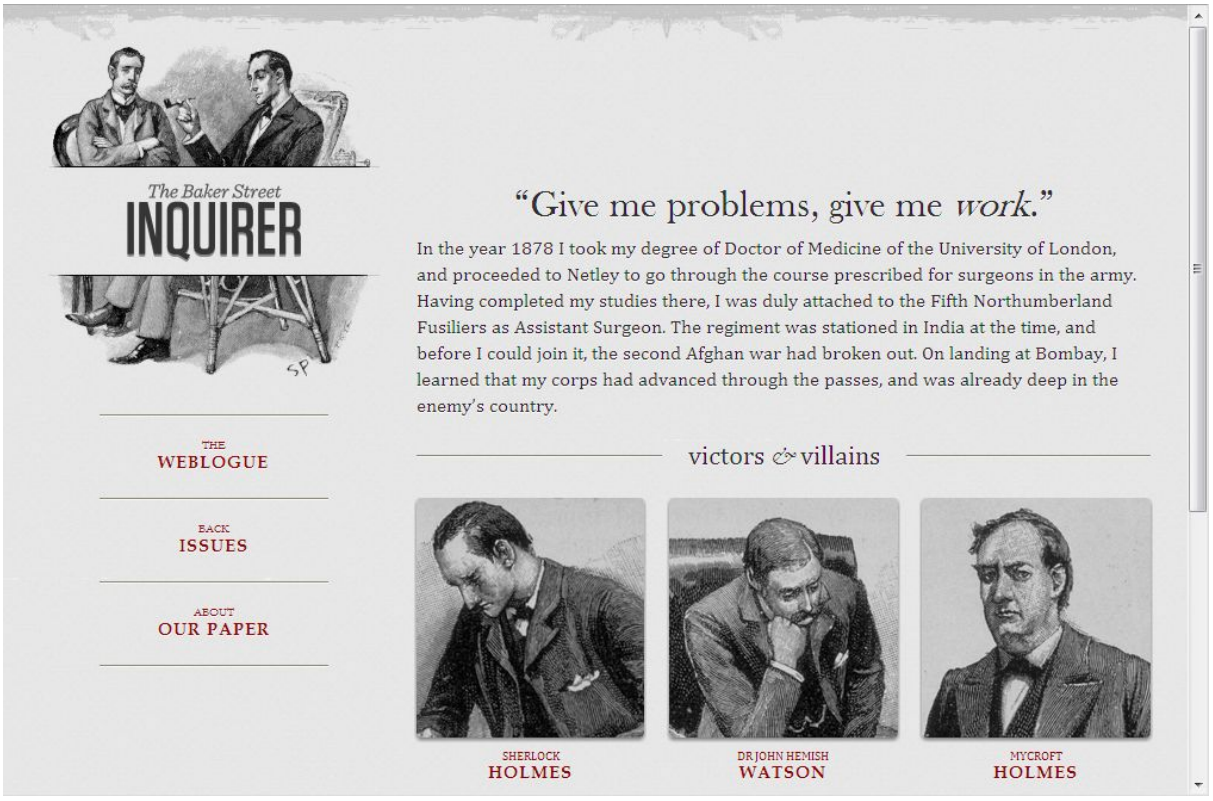
Aquí hay un ejemplo de sitio web adaptable: el sitio de Ethan Marcotte (gran referente de "Diseño adaptable") en Sherlock Holmes, donde dependiendo de la resolución, la pantalla difiere:



1360*768



1024*768



320*480

6. Ejemplo práctico

Ahora trataremos de poner en práctica lo que hemos visto. Para esto, mostraremos un sitio en tres columnas:

- la columna de la izquierda es el menú;
- la columna de la derecha es la visualización de anuncios;
- la columna central (la más importante) muestra listas de elementos.

Tomaremos como resolución de pantalla 1280x960 píxeles.

Comencemos con nuestro archivo HTML.

```
<!DOCTYPE HTML>
<html lang="es">
  <head>
    <title>Práctica de diseño adaptable</title>
          <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, maximum-scale=1, user-scalable=no" />
    <meta charset="utf-8" />

    <!-- Estilos -->
          <link type="text/css" media="screen" title="no title" rel="stylesheet" href="responsive.design.css" />
  </head>

  <body>
      <!-- Estructura de la página -->
      <div class="layout-main">
         <div class="main">
            <div class="layout-menu">
               <!-- Menú -->
               <div class="menu">
                  <h1>Menú</h1>
                  <ul>
                     <li><a href="#"Ir a menú 1</a></li>
                     <li><a href="#"Ir a menú 2</a></li>
                     <li><a href="#"Ir a menú 3</a></li>
                     <li><a href="#"Ir a menú 4</a></li>
                     <li><a href="#"Ir a menú 5</a></li>
                  </ul>
```

```
                    <select>
                         <option selected="selected">Seleccionar
un menú</option>
                        <option>Ir a menú 1</option>
                        <option>Ir a menú 2</option>
                        <option>Ir a menú 4</option>
                        <option>Ir a menú 5</option>
                    </select>
                </div>

            </div><div class="layout-content">
                <!-- Contenido -->
                <div class="content">
                    <div class="content-item">
                        <h1>Sobre Carl Sagan</h1>
                                    <p class="description">Breve
biografía</p>
                        <div class="article">
                                                <img src="image.png"
class="article-image" alt="Carl Sagan"/>

                                <p>Un personaje único e irrepetible
que nos acercó el universo a nuestros televisores. Carl Sagan
(1934 - 1996) fue un astrofísico, astrónomo y divulgador
científico estadounidense que, durante décadas, se dedicó a
explicar de forma sencilla todos los misterios que rodean al
cosmos.
También fue un pionero en la búsqueda de vida extraterrestre y
sus contribuciones fueron aplaudidas tanto por la comunidad
científica como por los telespectadores.</p>
                        </div>
                    </div>

                    <div class="content-item">
                        <h1>Sobre Carl Sagan</h1>
                             <p class="description">Sus frases más
famosas</p>
                        <div class="article">
                                                <img src="image.png"
class="article-image" alt="Carl Sagan" />
```

<p>En algún lugar, algo increíble está esperando a ser descubierto.

El nitrógeno presente en nuestro ADN, el calcio de nuestros dientes, el hierro de nuestra sangre, el carbono en las tartas de manzana… todos fueron creados en el interior de estrellas que chocaron entre sí. Estamos hechos del material de las estrellas.

En mi opinión, es mucho mejor entender el universo tal como es que persistir en el engaño, a pesar de que éste sea confortable.

Una reivindicación extraordinaria precisa de evidencias extraordinarias.

Somos como mariposas que vuelan durante un día pensando que lo harán para siempre.

Podemos juzgar el progreso por la valentía de las preguntas y la profundidad de las respuestas; por la osadía de encontrar la verdad más que en regocijarnos en lo que nos hace sentir bien.

El universo es un sitio bastante amplio. Si solo estamos nosotros, me parecería un auténtico desperdicio de espacio.

No quiero creer, quiero saber.

La belleza de la vida no hace referencia a los átomos que la componen, sino a la forma en que estos átomos se juntan.

El universo no parece ni hostil ni amigable, es simplemente indiferente.

La ausencia de evidencia no significa la prueba de la ausencia.

La ciencia es una forma de pensar, y no tanto un agregado de conocimientos.

La extinción es la regla. La supervivencia es la excepción.

Si algo puede ser destruido por la verdad, merece ser destruido.

La naturaleza es siempre más sutil, más compleja y más elegante que lo que somos capaces de imaginar.

Los átomos son, básicamente, espacio vacío. La materia está compuesta, principalmente, por la nada.

Un organismo que esté en guerra contra él mismo está condenado.

Vivimos en una sociedad extremadamente dependiente de la ciencia y la tecnología, en que casi nadie tiene unas mínimas nociones sobre ciencia y tecnología.</p>

```
                    </div>
                </div>
            </div>

        </div><div class="layout-adv">
            <!-- Publicidad -->
            <div class="adv">
                <div class="adv-item">
                    <img src="image.png" alt="Sagan" />
                    <br />
                    <a href="#">Enlace 1</a>
                </div>
              <div class="adv-item">
                    <img src="image.png" alt="Pub 2" />
                    <br />
                    <a href="#">Enlace 2 2</a>
                </div>

                <div class="adv-item">
                    <img src="image.png" alt="Pub 3" />
                    <br />
                    <a href="#">Enlace 3</a>
                </div>
    </div>
```

```
            </div>
          </div>
        </div>
    </body>
</html>
```

Seguidamente se muestra el código CSS:

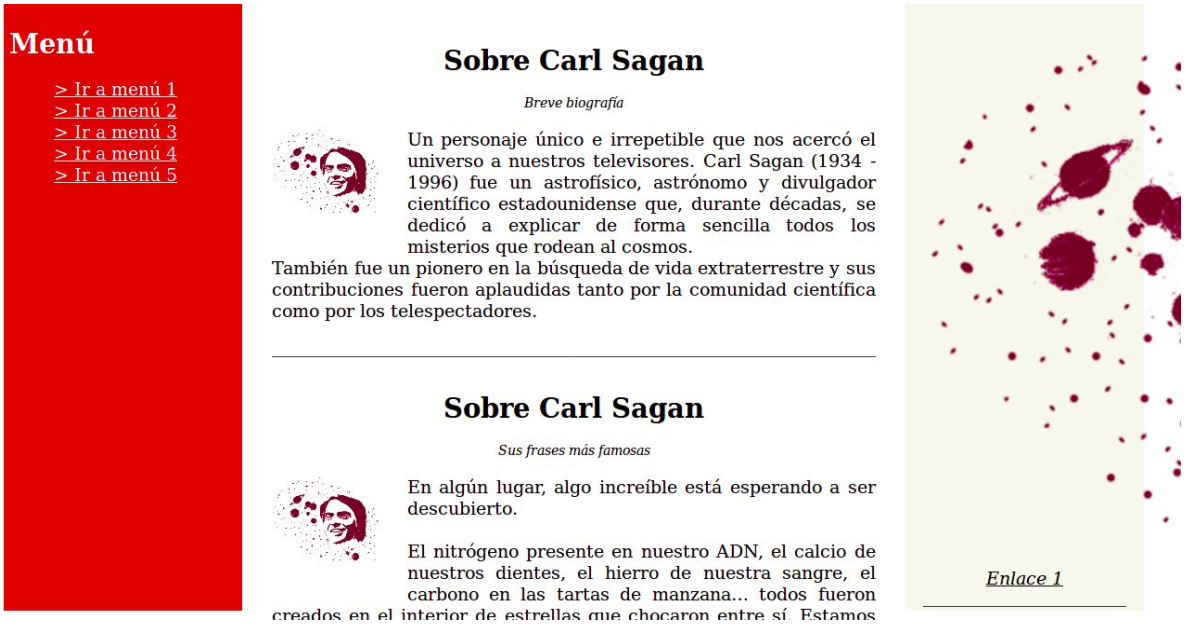
```
@charset "UTF-8";

/*
 Resolución ideal: 1280 * 960 pixels
 */

html, body {
    height: 100%;
    background-color: #FFFFFF;
    border: 0px solid transparent;
    font-size: 16px;
    margin: 0px 0px 0px 0px;
    min-height: 100%;
    padding: 0px 0px 0px 0px;
    width: 100%;
}

.layout-main {
    height: 100%;
    margin: 0px auto 0px auto;
    min-height: 100%;
    width: 1024px;
}

.main {
    height: 100%;
    min-height: 100%;
}

.layout-menu, .layout-content, .layout-adv {
    display: inline-block;
    height: 100%;
    min-height: 100%;
    vertical-align: top;
```

```
}

.layout-menu {
   color: #FFFFFF;
   background-color: #CC0000;
   width: 214px;
}

.layout-content {
   color: #000000;
   background-color: #FFFFFF;
   width: 596px;
}

.layout-adv {
   color: #000000;
   background-color: #F9F7ED;
   width: 214px;
}

.menu, .content, .adv {
   padding: 5px 5px 5px 5px;
}

.menu h1 {
   font-size: 24px;
}

.menu li {
   list-style: none;
}

.menu li a {
   color: #FFFFFF;
}

.menu li a:before {
   content: "> ";
}

.menu li a:hover {
   color: #000000;
```

```
}

.content .content-item {
   border-top: 1px solid #36393D;
   clear: both;
   margin: 0px auto 0px auto;
   padding: 15px 0px 15px 0px;
   text-align: center;
   width: 542px;
}

.content .content-item:first-child {
   border: 0px solid transparent;
}

.content .content-item h1 {
   font-size: 24px;
}

.content .content-item p.description {
   font-size: 12px;
   font-style: italic;
}

.content .content-item div.article {
   text-align: justify;
}

.content .content-item img.article-image {
   float: left;
   padding: 0px 25px 25px 0px;
   width: 96px;
}

.adv .adv-item {
   border-top: 1px solid #36393D;
   margin: 0px auto 0px auto;
   padding: 15px 0px 15px 0px;
   text-align: center;
   width: 182px;
}
```

```
.adv .adv-item:first-child {
   border: 0px solid transparent;
}

.adv .adv-item a {
   color: #000000;
   font-style: italic;
}
```

Luego obtenemos la siguiente pantalla:



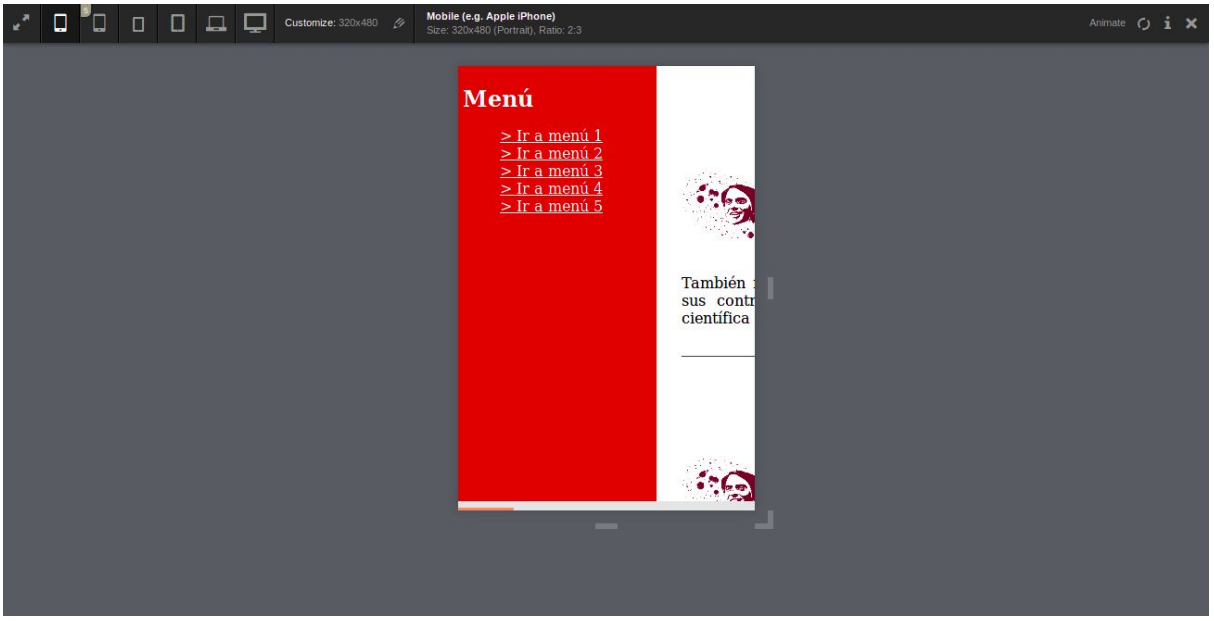
1280*960

La renderización es lo que se espera, para una resolución de 1280x960 píxeles. Pero quiero probar otras resoluciones (la extensión Window Resizer permite visualizar en el navegador a distintas resoluciones, https://chrome.google.com/webstore/detail/window-resizer/kkelicaakdanhinjdeammmilcgefonfh.

Comienzo probando a menos píxeles. Obteniendo esto:

Me doy cuenta de que tengo más y más espacios en los bordes de mi sitio, y que mi plantilla “está rota”. La optimización de la pantalla no está completa, y ya no es legible,.

Ahora tomo la resolución de un teléfono móvil:



Ya es ilegible y será peor y peor si bajamos la resolución.

6-A. Etapa 1

Comencemos con lo básico: establezcamos el estilo CSS que evita que los medios ocupen demasiado espacio, convirtiendo fuentes y haciendo la relación para los anchos. Obtenemos el siguiente estilo:

```
@charset "UTF-8";

/*
 Resolución ideal: 1280 * 960 pixeles
 */

html, body {
   height: 100%;
   background-color: #FFFFFF;
   border: 0px solid transparent;
   font-size: 1em; /* Ideal: 16px */
   margin: 0px 0px 0px 0px;
   min-height: 100%;
   padding: 0px 0px 0px 0px;
   width: 100%;
}

img, object, embed, canvas, video, audio, picture {
   max-width: 100%;
   height: auto;
}

.layout-main {
   height: 100%;
   margin: 0px auto 0px auto;
   min-height: 100%;
   width: 80%; /* Ideal: 1024px -> 1024 / 1280 */
}

.main {
   height: 100%;
   min-height: 100%;
}

.layout-menu, .layout-content, .layout-adv {
   display: inline-block;
```

```
    height: 100%;
    min-height: 100%;
    vertical-align: top;
}

.layout-menu {
    color: #FFFFFF;
    background-color: #CC0000;
    width: 20.8984375%; /* Ideal: 214px -> 214 / 1024 */
}

.layout-content {
    color: #000000;
    background-color: #FFFFFF;
    width: 58.203125%; /* Ideal: 596px -> 596 / 1024 */
}

.layout-adv {
    color: #000000;
    background-color: #F9F7ED;
    width: 20.8984375%; /* Ideal: 214px -> 214 / 1024 */
}

.menu, .content, .adv {
    padding: 5px 5px 5px 5px;
}

.menu h1 {
    font-size: 1.5em; /* Ideal: 24px */
}

.menu li {
    list-style: none;
}

.menu li a {
    color: #FFFFFF;
}

.menu li a:before {
    content: "> ";
}
```

```
.menu li a:hover {
   color: #000000;
}

.content .content-item {
   border-top: 1px solid #36393D;
   clear: both;
   margin: 0px auto 0px auto;
   padding: 15px 0px 15px 0px;
   text-align: center;
    width: 90.93959731543624161073825503355 7%; /*Ideal: 542px
-> 542 / 596 */
}

.content .content-item:first-child {
   border: 0px solid transparent;
}

.content .content-item h1 {
   font-size: 1.5em; /* Ideal: 24px */
}

.content .content-item p.description {
   font-size: 0.750em; /* Ideal: 12px */
   font-style: italic;
}

.content .content-item div.article {
   text-align: justify;
}

.content .content-item img.article-image {
   float: left;
   padding: 0px 25px 25px 0px;
    width: 16.107382550335570469798657718121%; /* Ideal: 96px
-> 96 / 596 */
}

.adv .adv-item {
   border-top: 1px solid #36393D;
   margin: 0px auto 0px auto;
```

```
   padding: 15px 0px 15px 0px;
   text-align: center;
    width: 85.04672897196261682242990654205 6%; /*Ideal: 182px
-> 182 / 214 */
}

.adv .adv-item:first-child {
   border: 0px solid transparent;
}

.adv .adv-item a {
   color: #000000;
   font-style: italic;
}
```

Si actualizo mi página en la resolución ideal, me doy cuenta de que nada se ha movido y si hago crecer o reducir la página, los elementos conservan una buena proporción. Acabo de completar parte de mi objetivo.

6-B. Paso 2

Es cierto que cuanto menor es la resolución, menos se puede explotar la pantalla. Especialmente porque mi ambición es ejecutar mi sitio en teléfonos inteligentes y tabletas, lo que implica resoluciones de pantalla bastante pequeñas.

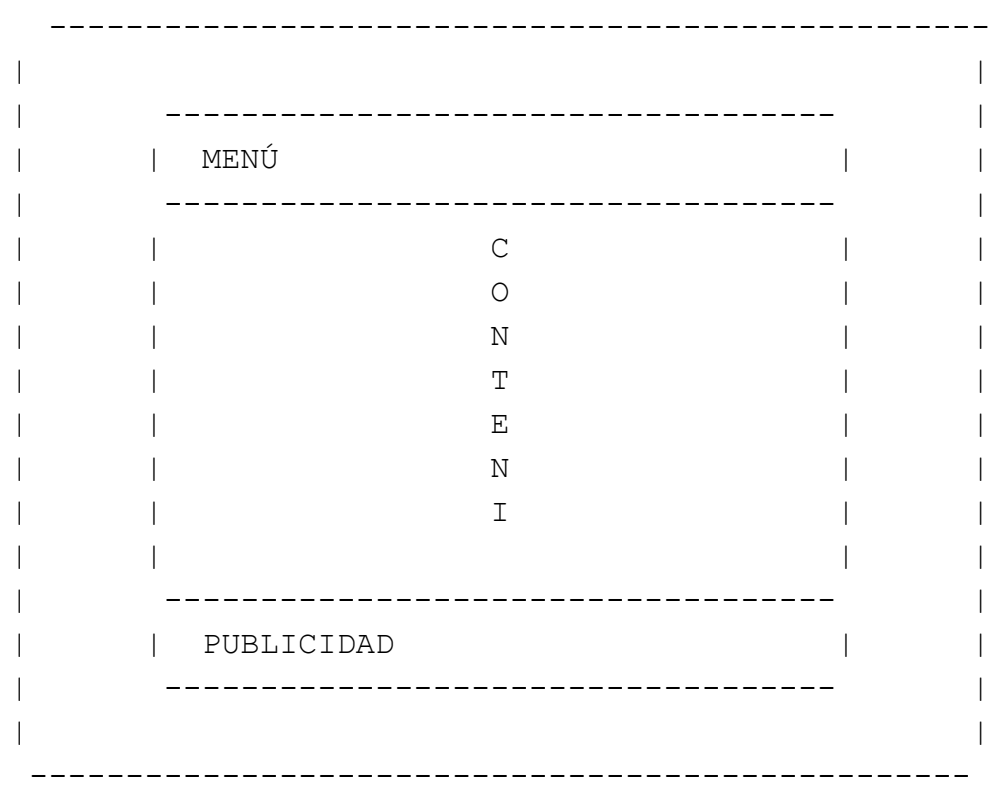
Por lo tanto, decido sobre la siguiente política:

- cuando alcanzo un ancho de menos de 1024 píxeles, reduzco las bandas blancas en los bordes;
- cuando llego a un ancho de menos de 960 píxeles, el menú va por encima de las otras dos columnas;
- cuando alcanzo un ancho de menos de 760 píxeles, no hay bandas blancas en los bordes;
- cuando alcanzo un ancho de menos de 640 píxeles, los anuncios se colocan debajo de los artículos;
- cuando llego a un ancho de 420 píxeles, Enmascaro los anuncios y se cambia el menú para que aparezca en una lista desplegable (lo que añade una etiqueta "<select>" en la página, los menús) ;
- cuando alcanzo un ancho de menos de 320 píxeles, ya no muestro la imagen que está en los artículos.

Si tuviéramos que representar eso, daría:

```
<!-- Estructure de nuestra página : -->
<!--
   Ideal :

    ------------------------------------------------
   |                                                |
   |       ----------------------------------       |
   |      |  M  |           C           |  P  |      |
   |      |  E  |           O           |  U  |      |
   |      |  N  |           N           |  B  |      |
   |      |  Ú  |           T           |  L  |      |
   |      |     |           E           |  I  |      |
   |      |     |           N           |  C  |      |
   |      |     |           I           |  I  |      |
   |      |     |           D           |  D  |      |
   |      |     |           O           |  A  |      |
   |      |     |                       |  D  |      |
   |       ----------------------------------       |
   |                                                |
    ------------------------------------------------

   Reformateado 1 :

     ------------------------------------------------
   |                                                |
   |       ----------------------------------       |
   |      |  MENU                           |       |
   |       ----------------------------------       |
   |      |                 C           |  P  |      |
   |      |                 O           |  U  |      |
   |      |                 N           |  B  |      |
   |      |                 T           |  L  |      |
   |      |                 E           |  I  |      |
   |      |                 N           |  C  |      |
   |      |                 I           |  I  |      |
   |      |                 D           |  D  |      |
   |      |                 O           |  A  |      |
   |      |                             |  D  |      |
   |       ----------------------------------       |
```

Reformateado 2 :



Reformateado 3 :

```
    |                                                    |
     ---------------------------------------------------

-->
```

Simplemente se ha de agregar al CSS ya hecho las siguientes *Media queries*:

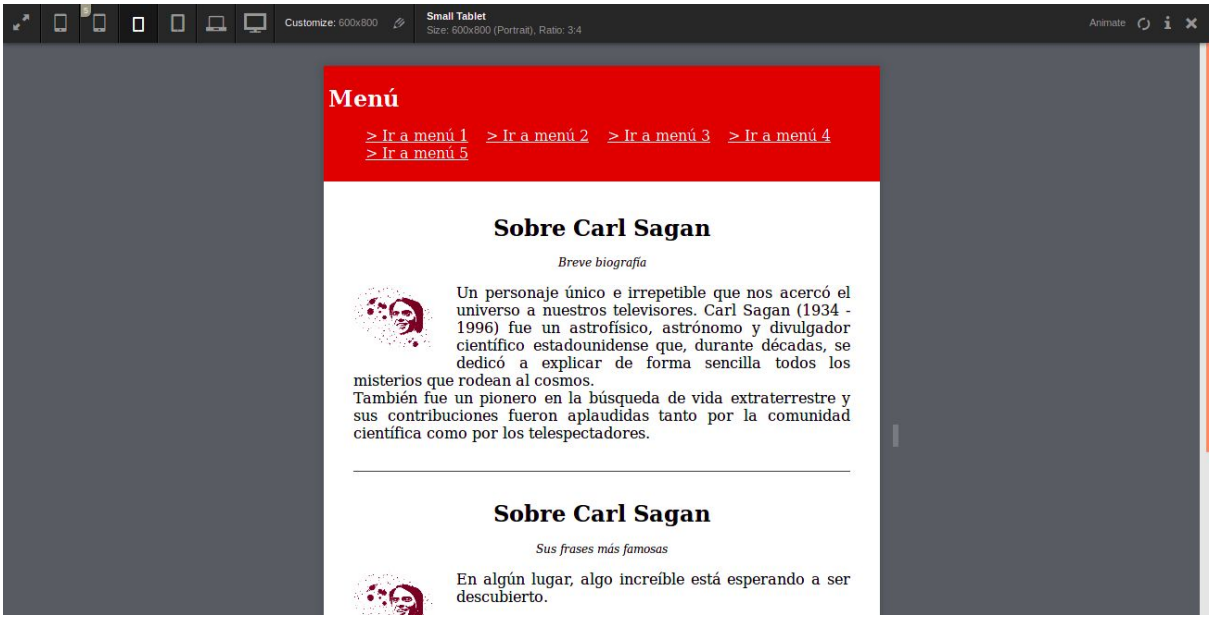
```
@media screen and (min-width: 421px) {
    .menu select {
        display: none;
    }
}

@media screen and (max-width: 1024px) {
    .layout-main {
        width: 95%;
    }
}

@media screen and (max-width: 960px) {
    .layout-menu {
        display: block;
        height: auto;
        min-height: auto;
        vertical-align: top;
        width: 100%;
    }

    .layout-content {
        width: 79.1015625%;
    }

    .menu li {
        display: inline-block;
            margin-right:  2.604166666666666666666666666667%;  /*
Ideal: 25px -> 25 / 960 */
    }
}

@media screen and (max-width: 760px) {
    .layout-main {
        margin: 0px 0px 0px 0px;
```

```
        width: 100%;
    }
}

@media screen and (max-width: 640px) {
    .layout-content {
        display: block;
        height: auto;
        min-height: auto;
        width: 100%;
    }

    .layout-adv {
        display: block;
        height: auto;
        min-height: auto;
        vertical-align: top;
        width: 100%;
    }

    .adv {
        text-align: center;
    }

    .adv .adv-item {
        display: inline-block;
    }
}

@media screen and (max-width: 420px) {
    .menu ul {
        display: none;
    }

    .menu select {
        display: inherit;
        width: 96%;
    }

    .layout-adv {
        display: none;
    }
```

```
}

@media screen and (max-width: 320px) {
    .content .content-item img.article-image {
        display: none;
    }
}
```

Esto genera las siguientes pantallas:

Tableta



Teléfono grande

Teléfono

V6-C. Paso 3

Ahora, solo resta hacerlo funcionar en nuestro enemigo endémico: IE. Para hacer esto, simplemente agrega en el encabezado de la página HTML "respond.js":

```
<head>
   <title>Responsive design</title>
         <meta   name="viewport"   content="width=device-width,
initial-scale=1,        minimum-scale=1,        maximum-scale=1,
```

```
user-scalable=no" />
  <meta charset="utf-8" />

   <!-- Styles -->
      <link type="text/css" media="screen" title="no title"
rel="stylesheet" href="responsive.design.css" />

   <!-- JavaScripts -->
         <script charset="utf-8" type="text/javascript"
src="respond.min.js"></script>
</head>
```

Hay sorpresas en IE6 e IE7: la propiedad «display: inline-block» está poco reconocida. Sea lo que sea, tengo una utilidad de CSS para estos navegadores:

```
/*
 Hack IE6 CSS
 */
.content .content-item {
   _border-top: expression(this.previousSibling == null ? "0px
solid transparent" : "1px solid #36393D");
}

.adv .adv-item {
   _border-top: expression(this.previousSibling == null ? "0px
solid transparent" : "1px solid #36393D");
}

/**
 Hack IE6-7 CSS
 */
.layout-main {
   *clear: both;
}

.layout-menu, .layout-content, .layout-adv {
   *display: block;
   *float: left;
}

@media screen and (max-width: 960px) {
   .layout-menu {
```

```
        *float: none;
    }

    .menu li {
        *display: block;
        *float: left;
    }
}

@media screen and (max-width: 640px) {
    .layout-adv {
        *float: none;
    }

    .layout-content {
        *float: none;
        *width: 100%;
    }
}
```

Y obtenemos el efecto esperado en IE8, pero también en IE7 e IE6.

7. ¿Esto es todo?

Al aplicar reglas simples, logramos obtener un sitio adaptable, que tiene la ventaja de ejecutarse en un gran conjunto de navegadores y versiones de navegador.

Sin embargo, ser adaptable no solo está del lado de nuestros navegadores. Aunque tratamos de mostrar todos los elementos si ofrecemos la funcionalidad y si tratamos de importar todos los recursos, obtendremos sitios muy devoradores y una experiencia del usuario molesta.

Un sitio web adaptable también es un sitio web que solo importará recursos cuando sea necesario (no es necesario importar rellenos en teléfonos inteligentes y tabletas: las funciones ya están allí).

También es un sitio web que mostrará imágenes adaptadas (se puede proponer una resolución y / o imagen de compresión diferente para teléfonos inteligentes y tabletas, reduciendo así el ancho de banda).

Se pueden ofrecer las características esenciales de acuerdo con el espacio disponible (si tenemos más cosas para mostrar, será más difícil visualizarlas con una resolución pequeña, también se aligera la página).

En pocas palabras, hacer diseño adaptable es ofrecer un sitio web que se adapta al soporte (usar madera para construir la Torre Eiffel sería laborioso y emplear zinc para una puerta de entrada un poco ambicioso).

8. Carga dinámica y condicional en el lado del cliente

Ser adaptable significa asegurarse que el sitio es más rápido en la carga y en su tiempo de ejecución. Para ello cargaremos hojas de estilo y JavaScript solo cuando sea necesario.

Según Google, "la velocidad es una característica". En otras palabras, la velocidad es una característica en sí misma. Por lo tanto, su objetivo es cargar (por ejemplo) Gmail en menos de dos segundos, incluso si el buzón de correo contiene varios GB de correos electrónicos. Por supuesto, no cargan todos los correos electrónicos.

En primer lugar, se recuperarán los archivos JavaScript solo cuando se cargue la página HTML, esto permite tener una visualización inmediata y la descarga/interpretación de los archivos JavaScript será más rápida. Hay entornos de trabajo que nos ayudan a hacer esto, como "RequireJs", donde declaramos módulos (e incluso dependencias entre módulos), http://requirejs.org/.

Entonces, Gmail sólo importa JavaScript al mínimo. El procedimiento es muy sencillo: cada vez que se utiliza una característica por primera vez, se lanza una petición AJAX para cargar el JavaScript correspondiente, interpretarlo y luego ponerlo a disposición. Esto permite que la aplicación consuma memoria solo cuando sea necesario. Y por lo tanto hacerlo menos codicioso que si tuviéramos que cargarlo todo. Con "RequireJS", esto también se hace de forma sencilla.

Luego, para CSS, normalmente lo colocamos en la parte superior de nuestra página HTML, pero podemos usar las "media queries" para cargarlo solo cuando sea necesario, como se ilustra en el siguiente ejemplo:

```
<link rel="stylesheet" media="screen" href="layout.css" />
<link rel="stylesheet" media="screen and (max-width: 640px)"
href="small-screen-layout.css" /><!-- Media Queries -->
<link rel="stylesheet" media="screen and (min-width: 1240px)"
```

```
href="large-screen-layout.css" /><!-- Media Queries -->
<link rel="stylesheet" media="print" href="print.css" />
```

9. Iconos y fuentes independientes de la resolución

Hemos visto antes cómo mostrar los *Media queries* correctamente y asegurarnos que la página siempre tenga las mejores proporciones. Pero no nos interesan las imágenes que usamos (usamos la propiedad CSS "background-image"), ni qué fuentes se ajustan mejor.

Para nuestra imagen de fondo, esto es muy importante, porque cada vez aparecen más las llamadas "pantallas retina" (especialmente en las tabletas). Esto significa que la densidad de píxeles (dpi) cambia. En otras palabras, un píxel a veces corresponde... a dos/tres píxeles físicos. Esto significa que una imagen con un dpi normal se mostrará más pequeña en una pantalla "retina". Tenemos dos técnicas para salir de esto.

Utilizando *Media queries* que, dependiendo de los dpi, utilizará la mejor imagen:

```
a.icon {
    background-image:url(images/icon.png);
}

@media only screen and (-webkit-min-device-pixel-ratio: 1.5),
    only screen and (-o-min-device-pixel-ratio: 3/2),
    only screen and (min--moz-device-pixel-ratio: 1.5),
    only screen and (min-device-pixel-ratio: 1.5) {

    a.icon {
        background-image:url(images/icon@2x.png);
    }
}
```

O usando conjuntamente la propiedad CSS3 background-size (permitiendo redimensionar una imagen declarada en background-image) y usando un SVG. Como resultado, dispondremos de una imagen vectorial que se adaptará correctamente a cualquier tipo de pantalla. Nada nos impide, en cuanto a nuestras imágenes clásicas, tener SVG conteniendo *sprites* (un conjunto de iconos reunidos en una sola imagen, para evitar cargar varios archivos).

```
a.icon {
    font-size: 1em;
```

```
    background-image: url('icon.svg');
    background-size: 1.875em 10em;
}
```

Por supuesto, ambas propuestas están basadas en HTML5/CSS3. Para navegadores más antiguos, podemos utilizar el entorno de JavaScript "RetinaJS", http://retinajs.com/.

Para las fuentes, es posible utilizar la propiedad "@font-face" de CSS3 para declarar las fuentes (y especialmente los recursos relacionados) que se pueden adaptar de acuerdo con la resolución (especialmente teniendo fuentes basadas en SVG).

```
@font-face {
    font-family: 'Icon Font';
    src: url('icon-font.eot');
    src: local('...');
    url('icon-font.woff') format('woff'),
    url('icon-font.ttf') format('truetype'),
    url('icon-font.svg') format('svg'); }

.icon {
    font-family: 'Icon Font';
    font-size: 20px;
}
```

Puedes ver un ejemplo de ello en el entorno "Fontello", http://fontello.com/.

10. Imágenes comprimidas

También hay técnicas para tener imágenes con un tamaño de archivo muy pequeño, pero que se pueden mostrar correctamente, sin efecto de desenfoque o pixelado.

El principio es muy simple: si tenemos una imagen con una resolución muy alta y una relación de compresión muy alta, su tamaño será menor que la misma imagen dos veces menor con compresión normal.

Como lo ilustran en "Smashing Magazine" sobre “Responsive Design", https://speakerdeck.com/smashingmag/responsive-web-design-clever-tips-and-techniques , podemos obtener una ganancia del 70% en el tamaño de las imágenes

11. RESS

Hacer "Diseño Adaptable" significa establecer una nueva arquitectura a la que no estamos necesariamente acostumbrados. A menudo, nos limitamos a la parte del navegador, pero la parte del servidor no debe ser descuidada. Este es el principio "RESS" (*Responsive Server-Side*).

Para ello disponemos de marcos de referencia. Podemos citar el proyecto "Wurfl" de Facebook, pero también el proyecto OpenSource "ua-parser" (https://github.com/tobie/ua-parser) que nos permite detectar si estamos en un soporte móvil o no, o incluso detectar las capacidades de los navegadores.

Así, podemos imaginarnos ofrecer una interfaz adaptada en función del soporte: un CSS diferente, un *widget* cuya visualización/uso se adapte al soporte, genere su página de forma diferente no incluyendo los anuncios, los comentarios de la página....

Esto no significa que la detección en el lado del navegador sea inútil, estoy diciendo que no debemos limitarnos a uno u otro: los dos son complementarios. Inicialmente podemos utilizar sólo uno de los dos, pero muy rápidamente alcanzaremos los límites o dificultades/ventajas que la otra solución podría proporcionarnos.

Tomemos el caso de los comentarios y anuncios: podríamos muy bien colocarlos en la página y ocultarlos dependiendo del tamaño de la pantalla con *Media queries* de CSS3. Pero generaremos HTML/JavaScript para nada y especialmente consumiremos tráfico de red innecesario.

Podemos imaginar que si vemos que nuestro soporte es móvil, no generamos el HTML correspondiente, de lo contrario lo ocultamos, y lo mostramos de nuevo vía CSS3. O bien, generamos imágenes que se adaptan a los móviles (como pudimos ver en el punto anterior).