Diseño Web Responsive
Tutorial con ejemplos adaptables

Para hacer una página adaptable se usa la técnica *Media queries*, que nos permite condicionar varios estilos dependiendo la resolución de la pantalla.

En los comienzos de la web, todas las páginas eran “rígidas", es decir, de ancho fijo en pixeles. En los años 90, los monitores de 14 pulgadas con un ancho de pantalla de 640px eran los únicos que existían, y ese era el tamaño que se daba a los diseños. Simplemente se definían tamaños en pixeles a todas las columnas.

Cerca del año 2000, surgió un segundo valor a considerar: 800px de ancho (ya existían monitores de 15 y de 17 pulgadas). El problema era que, si definíamos el ancho total de nuestro sitio en 800px, a los usuarios de monitores de 640px de ancho que todavía quedaban, les aparecía una poco elegante y nada usable barra de desplazamiento horizontal. El sitio no entraba en la pantalla.

Las soluciones para ese escenario dividido en dos tamaños eran muy sencillas: hacíamos un contenedor general de 640px de ancho y lo centrábamos dejando una pequeña franja sin ocupar a ambos costados o bien hacíamos un sitio con ancho definido en porcentajes, que se estiraba ligeramente entre ambas medidas.

Más cerca en el tiempo, alrededor del año 2005, la decisión a tomar era similar: seguían existiendo monitores de 800px (ya no quedaba ningún monitor de 640px de ancho), pero la resolución mayoritaria pasó a ser de 1024px. Entonces, o le dábamos ancho ***líquido*** (en porcentajes) para que se estirara entre 800 y 1024px, o creábamos un contenedor de 800px centrado, sobrando unas pequeñas franjas a los costados, y asunto resuelto.

Pero al llegar la década del 2010, un cambio verdaderamente revolucionario alteró nuestro escenario. Los teléfonos móviles empezaron a navegar la web, usando navegadores similares a los de un PC (Opera, Safari, Explorer, Chrome y Firefox) y apareció la primera tablet (el iPad). Desde ese momento, nuestras decisiones de diseño se volvieron más complejas, a causa de la diversidad de tamaños de pantalla con los que los usuarios navegan nuestros sitios. ¿Para cuál de todos esos tamaños vamos a diseñar? ¿Para todos? ¿Eso es posible?

Página adaptable a distintos dispositivos

Algún cambio es necesario en nuestra metodología de diseño, ya que un diseño líquido sencillamente no soporta ser “estirado" desde un mínimo de 320px hasta varios miles de pixeles de ancho.

Y por otro lado, si en lugar de darle ancho líquido, le definiéramos los anchos en pixeles, el efecto sería igual de negativo: el sitio se “miniaturizará", siendo muy incómodo navegarlo, dependiendo del uso constante de zoom en ambos sentidos y debiendo realizar desplazamientos horizontales. Algo así como sobrevolar la página web en helicóptero, a la manera de cómo visualizamos un mapa.

Si el sitio tiene ancho fijo, se miniaturiza y dependemos de hacer zoom constantemente, algo muy poco usable.

En el escenario actual, ya no podemos confiar en una resolución mayoritaria, ni en dos o tres, sino que es preciso comprender que las resoluciones utilizadas por los usuarios cada vez estarán más fragmentadas y serán más variadas, cubriendo

rangos intermedios entre dos puntas cada vez más distantes: dispositivos más pequeños que un monitor de PC, por un lado, y pantallas cada vez con mayor resolución en el otro extremo.

La era en la que usamos y abusamos de sitios de ancho fijo en pixeles, rejillas y columnas definidas en pixeles y hasta metodologías de diseño visuales como las de maquetar usando celdas de tablas, ya llegó a su fin. Es el fin del diseño rígido. Necesitamos algo nuevo.

Diseño Web Responsivo. Fundamentos para crear sitios adaptables.

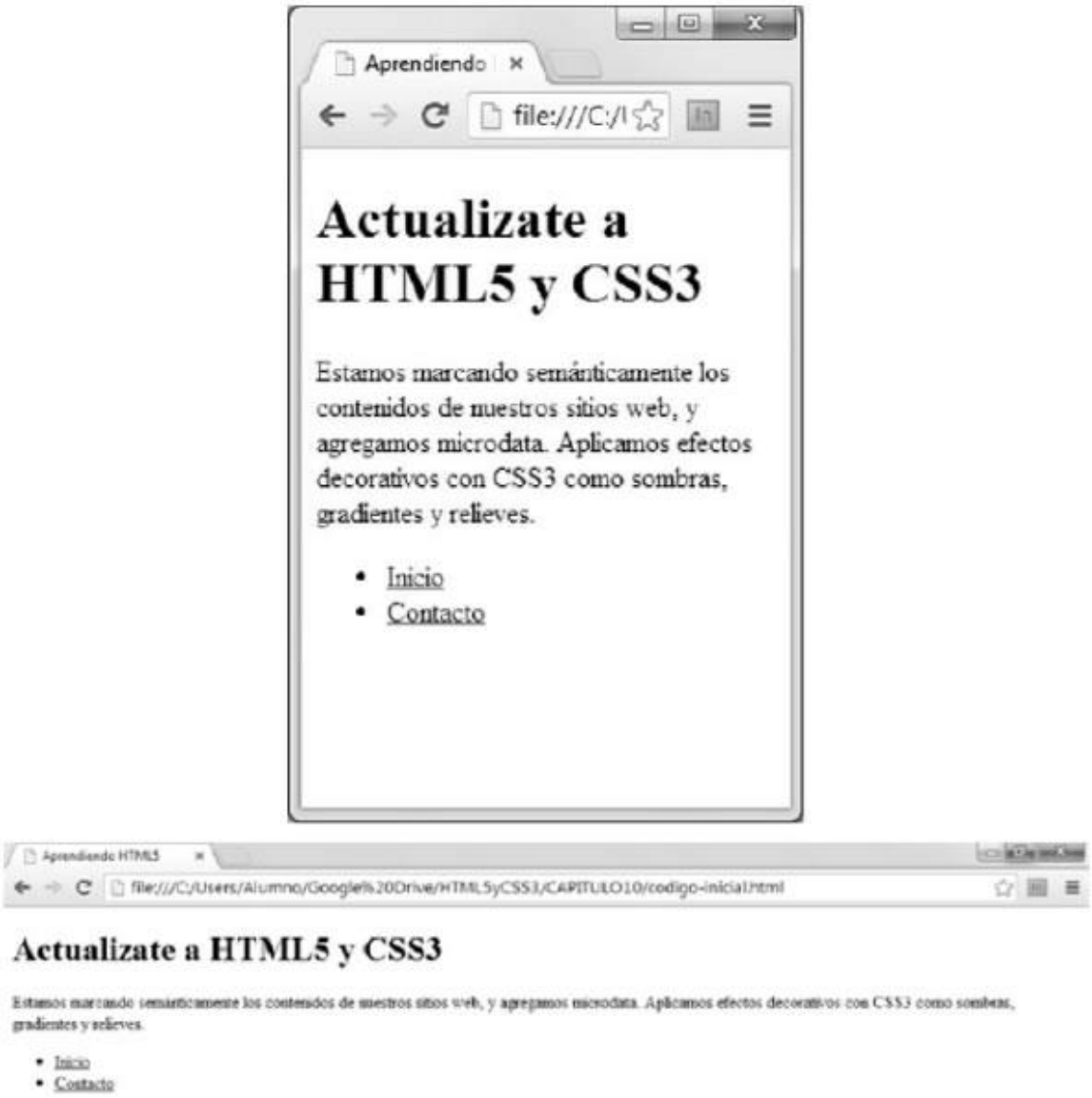
Para hacer una página adaptable se usa la técnica ***Media queries***, que nos permite condicionar varios estilos dependiendo la resolución de la pantalla.

El diseño como problema y como solución

Las páginas web por naturaleza son “adaptables" desde el nacimiento mismo de la Web. Para comprobar que esta hipótesis es correcta, pensemos en un archivo HTML correctamente estructurado, con encabezados, párrafos y listas. Pero sin diseño. Es decir, un archivo HTML al que no le vinculamos ninguna hoja de estilos, al que no lo maquetamos. Por ejemplo, el siguiente:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Aprendiendo HTML5</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Actualizate a HTML5 y CSS3</h1>
        <p>Estamos marcando semánticamente los contenidos de nuestros sitios
      web, y agregamos microdata. Aplicamos efectos decorativos con CSS3 como
      sombras, gradientes y relieves.</p>
        <ul>
            <li><a href="index.html">Inicio</a></li>
            <li><a href="contacto.html">Contacto</a></li>
        </ul>
    </body>
</html>
```

Si tratamos de navegar este archivo desde dispositivos de distintos tamaños, no tendremos ningún problema para acceder a sus contenidos: el texto será legible en un móvil, en una tablet y en un PC también.



Un HTML sin estilos (sin diseño) es adaptable por naturaleza, es adaptable porque se puede usar en todos los tamaños.

En conclusión, el problema que dificulta acceder a nuestros sitios web desde móviles es generado por nuestro proceso de diseño: surge cuando agregamos hojas de estilo y diseñamos, dando medidas a cada bloque de contenidos y a los elementos que lo componen (imágenes, tipografías, etc.).

Aunque más que el diseño en sí, diríamos que es el “sobre-diseño” el culpable de la rigidez. El error consiste en definir toda medida en pixeles, pretendiendo controlar al detalle la ubicación de la información. Ese es el principal causante de esta inflexibilidad.

Es decir, somos los diseñadores mismos los que causamos el problema, con nuestro antiguo proceso de diseño (heredado de medios gráficos como el papel) y la falla está en algunas decisiones que tomamos en su transcurso (como las unidades de medida que elegimos aplicar).

> El hecho de que seamos los responsables del problema es una muy buena noticia, ya que si estamos provocando el problema, también está en nuestras manos la solución.

No depende de “otros”, no es causada por los fabricantes de dispositivos ni de navegadores ni de editores HTML: está plenamente en nuestras manos la decisión de hacer que nuestros sitios empleen técnicas adaptables.

¿Qué son las Media Queries?

Ante todo, la técnica: se denomina Media Queries a la posibilidad de definir condiciones que deben cumplirse para que el navegador procese algunas reglas de estilo específicas dentro de nuestra hoja CSS.

Por ejemplo, podríamos definir una condición que determine que si el tamaño de pantalla es menor a 480px de ancho, el navegador procese ciertos estilos CSS que dejaremos preparados. En esos estilos, definiremos diseños ideales para un dispositivo que cumpla con esa condición, es decir, que no supere ese ancho de pantalla de 480px.

```
@media screen and (max-width:480px){
    body {
        font-size:80%; // achicamos la fuente en pantallas pequeñas
    }
}
```

Mediante el uso de esta técnica, se intenta optimizar la experiencia de uso de un sitio web para adaptar el diseño al contexto. En este caso, sería un tamaño variable de pantallas el que le brinda un contexto muy particular a los contenidos, limitando los diseños posibles.

Combinación de condiciones para crear rangos

También podemos combinar varias condiciones en una sola utilizando el operador lógico “and” tantas veces como sub-condiciones necesitemos combinar. En el siguiente ejemplo, combinaremos dos condiciones: que sea una pantalla y que mida como mínimo 1024 pixeles de ancho.

```
@media screen and (min-width: 1024px) { }
```

En este otro ejemplo, combinaremos tres condiciones: que sea pantalla, que mida más de 320px y menos de 768 pixeles de ancho.

```
@media screen and (min-width: 321px) and (max-width: 768px) { }
```

Deberemos tener especial cuidado en que los valores de los rangos que definamos no se superpongan. Es decir, si el valor máximo de uno es 320, definiremos el inicio del siguiente rango en 321 y no en 320 exactos, ya que de lo contrario se leerán las dos hojas cuando un dispositivo tenga exactamente esa medida.

Esta posibilidad de combinar varias condiciones será indispensable para poder detectar los rangos de tamaños a los que se aplicará una u otra hoja de estilos.

Diseño web adaptable/sensible/responsive

Esta técnica llamada media queries, que nos permite condicionar la lectura de distintos estilos, es la que nos abre la posibilidad a un nuevo enfoque de diseño web, que resulta ideal para dar soporte a la enorme diversidad de dispositivos existentes actualmente. Este enfoque es conocido como “diseño web adaptable” o “diseño web sensible” (**responsive web design**, originalmente).

En pantallas grandes (una media querie de “Mayor a 1024px”, por ejemplo) podríamos hacer flotar 3 columnas, mientras que en una tablet (una media querie de “Mayor a 800px”, por ejemplo) solo dejaremos flotar 2 columnas, y en el caso de pantallas más estrechas (móviles, sin media queries) anularemos todos los flotados, reduciendo la cantidad de columnas a una sola (“apilando” los contenidos uno debajo del otro).

Veamos cómo sería el código:

```
/* Diseño para móviles */
#columna1, #columna2, #columna3 {
    float:none;
    width:100%;
}

/* Diseño para tablets */
@media screen and (min-width: 800px) {
    #columna1, #columna2 {
        float:left;
        width:50%;
    }
}

/* Diseño para PC */
@media screen and (min-width: 1025px) {
    #columna1, #columna2, #columna3 {
    float:left;
    width:33%;
}
```

Un código similar al anterior es el utilizado para lograr estos cambios en el sitio del Japan Times:

La zona sin media queries de la hoja CSS aplica en móviles, la primera media query aplica en tabletas (dos columnas) y la segunda media query aplica en PCs y flota las 3 columnas.

Pero no solo la disposición puede cambiar entre una pantalla y otra; analizaremos a continuación cuáles son los elementos de diseño que es importante adaptar para que el contenido web quede optimizado para diferentes dispositivos.

Elementos del diseño web adaptable

Los elementos centrales de un diseño adaptable o “sensible” son al menos cuatro:

1. Esquema tipográfico flexible
2. Maquetación adaptable usando Media Queries, modificando la cantidad de columnas del diseño, con rejillas flexibles, adaptando los anchos dentro de cada rango.
3. Imágenes y multimedia adaptables, cambiando el tamaño de las imágenes y otros elementos vinculados. No debemos olvidar, si el sitio los incluye, la adaptación de videos y animaciones, y demás contenidos complejos como mapas, tablas, slides, etc.
4. Navegación adaptable, optimizada para touch en pantallas pequeñas.

Complementariamente, aplicaremos sistemáticamente en todas nuestras páginas adaptables, dos elementos extra:

a. Una etiqueta meta viewport que impida la “miniaturización” de nuestras páginas.
b. Y un script compatibilizador para que funcione en navegadores antiguos la técnica de media queries.

Comencemos a analizar uno por uno estos elementos, así aprenderemos a incluirlos en nuestros proyectos, para que podamos volverlos adaptables y dejemos atrás la era de la rigidez.

1. Esquema tipográfico flexible

El primer elemento del diseño que volveremos flexible desde nuestra hoja de estilos será el texto. La novedad será que cambiaremos la unidad de medida más utilizada hasta el momento para definir el tamaño de las fuentes mediante la propiedad font-size (es decir, los pixeles), por otras dos unidades de medida más versátiles.

Si continuamos utilizando pixeles como unidad de medida para nuestras fuentes, tendremos dos tipos de problemas:

1. El primero se lo causaremos a algunos de nuestros usuarios, aquellos que utilicen Internet Explorer, ya que no podrán escalar el tamaño de la fuente si ésta fue definida en pixeles (un problema de accesibilidad que afectará a nuestros usuarios miopes).
2. Y el segundo, será un problema de mantenimiento para nosotros mismos, ya que deberíamos duplicar, triplicar o más, las declaraciones de tamaños de fuentes dentro de cada zona de la hoja de estilos, para poder definir cuál debe ser su tamaño en cada rango o condición, es decir, en cada media query. Entenderemos este problema a continuación.

Texto primero

¿Por dónde empezamos entonces el proceso de codificación (HTML) si queremos aplicar Media Queries? Por el principio: el texto.

Pregunta al paso que siempre me gusta plantear a mis alumnos: ¿Cuál es el denominador común de una pizza? ¿Aceitunas? ¿Cebollas? ¿Tomates? ¿O la masa?

¿Y el denominador común de una Web? ¿Flash? ¿Imágenes? ¿Videos? ¿O el texto?

Una vez marcados los textos semánticamente (con h1, h2, p, ul con li), aplicamos los identificadores y clases que creamos necesarios, y ya tendremos el código HTML básico terminado. Ese texto bien estructurado con HTML elemental, será suficiente para cualquier móvil de bajas prestaciones que aún pudiera estar utilizando algún usuario.

Tipografía, siempre en EM

Ahora sí, llegó el momento de definir en nuestra hoja de estilos las unidades de medida que volverán flexible nuestro esquema tipográfico, para garantizar la legibilidad de nuestros contenidos. Pensemos que la distancia de lectura en un PC es cercana al metro, mucho mayor a la que existe entre nuestros ojos y un teléfono o tablet que sostenemos en nuestras manos:

Distancia de lectura y tamaño de fuente comparado.

Por ese motivo, los tamaños de fuentes deberán ser mayores para un PC que para una tablet, y los de una tablet, mayores que los de un móvil.

Para lograrlo, no usaremos más pixeles para definir el tamaño de fuentes, sino que usaremos la unidad de medida em y los porcentajes, combinados de una manera particular: al body (y solamente al body) le definiremos un font-size base en porcentaje, y al resto de textos, se lo definiremos en em:

```
body {
    font-size:80%;
}

/* porcentaje base, solo en el body */
p {
    font-size:0.9em;
}
h1 {
    font-size:2em;
}
h2 {
    font-size:1.4em;
```

```
}
#pie {
    font-size:1.2em;
}
.epigrafes {
    font-size:1.1em;
}

/* fin de zona común a todas las resoluciones */
@media screen and (min-width:800px) {
    body {
        font-size:100%;
        /* ampliamos los textos si mide más de 800px */
    }
}

/* fin de la zona para más de 800px de ancho de pantalla */
@media screen and (min-width:1200px) {
    body {
        font-size:120%;
        /* ampliamos más aún los textos si mide más de 1200px */
    }
}

/* fin de la zona para más de 1200px de ancho de pantalla */
```

Observemos que hemos definido en em los tamaños de fuentes de nuestros contenidos usando indistintamente etiquetas, id o clases, en un solo lugar: la zona inicial de nuestra hoja de estilos, aquella que leerán todos los dispositivos sin condiciones, ya que no se encuentra envuelta en ninguna media query.

Por otro lado, cuando aplicamos una condición para cierto tamaño, lo único que cambiamos es el valor base en porcentaje aplicado al body, lo que hará que todo el esquema tipográfico definido en ems se escale proporcionalmente a un nuevo tamaño. Es decir, lo que estamos haciendo es cambiar el tamaño del em, al cambiar su punto de referencia (que es ese porcentaje que definimos en el body).

Como orientación, podemos calcular que en la mayoría de las pantallas de PC, a tamaño de fuente normal, la equivalencia entre ems y pixeles es que 1em = 16px, por lo que, si queremos definir en la hoja de estilos un texto que en Photoshop ocupaba 24px, lo dividiremos por 16 para saber su valor en em, que en este caso sería 1.5em. *Atención: utilicemos puntos como separador decimal, no comas.*

Por supuesto, nada impide que realicemos ajustes en alguna de las media queries si queremos que determinado texto tenga una medida especial en una de ellas. Pero el punto de partida ya quedará establecido automáticamente.

Con estas nuevas unidades de medida ya podremos crear esquemas tipográficos adaptables, que se visualicen de manera óptima según la distancia de lectura de cada dispositivo.

Notemos que no estamos definiendo un tamaño rígido, sino una relación proporcional entre los distintos tamaños de texto de nuestra página, proporción que

se mantendrá a lo largo de todos los dispositivos gracias a que vamos a escalar el esquema completo en cada media query.

2. Maquetación adaptable y rejilla flexible

Aprovechando que cada zona de la hoja de estilos que envolvamos entre media queries permite que ciertos estilos se apliquen solo en un rango de tamaños de pantalla, en cada zona acomodaremos los contenidos en columnas de una manera optimizada para el tamaño del dispositivo.

"Flotar o no flotar, that is the question"

En principio, convengamos que en un móvil básico el espacio no es suficiente para flotar dos o más columnas una al lado de la otra, por lo que la disposición será extremadamente simple: solo dejaremos que los bloques se apilen uno debajo del otro por orden de aparición en el código HTML.

A partir de allí, haremos flotar tantos bloques como creamos necesario en cada media query.

Rejillas flexibles

La rejilla flexible consiste en definir anchos de contenedor y de columnas en porcentajes, para que los bloques de un diseño mantengan una proporción entre sí dentro del rango de ancho mínimo y máximo definido en cada media query del punto anterior.

La fórmula para calcular los porcentajes a partir de un boceto en pixeles es la de dividir el ancho del elemento por el del que lo envuelve:

Tamaño Elemento / Tamaño Contexto

Ejemplo: 600px / 960px = 0,625

Es decir, ese bloque que medía 600px dentro de un contenedor de 960px ahora deberá medir 62.5% (ese valor surge del 0.625 de la cuenta que acabamos de realizar)

Repitamos la fórmula en otro caso:

Tamaño Elemento / Tamaño Contexto

Ejemplo de 3 columnas para fotos ubicadas dentro de una zona de 480px:

150px / 480px = 0,3125

Es decir, deberemos definir un 31.25% de ancho a cada una de las 3 columnas.

Márgenes y paddings flexibles

En este contexto, también debemos volver flexibles los márgenes y paddings, para que no arruinen con pixeles la proporcionalidad de los espacios conseguida.

1. Margen: el contexto es el ancho del elemento padre (contenedor):
    - Ej. 24px / 900px = 0,02666 = 2,66%

2. Padding: el contexto es el ancho del elemento mismo al que lo aplico:
    - Ej. 24px / 200px = 0,12 = 12%

3. Medios adaptables (imágenes, videos)

Si en cada zona de los estilos CSS delimitada por una media querie apuntamos a distintas imágenes (de distintos tamaños), no tendremos problemas con background-image:


```
/* La imagen más pequeña va primero, sin condiciones, porque es para el móvil más pequeño */
.logo {
    background-image: url(logo-chico.jpg);
}

/* Logo mediano, para anchos de pantalla de entre 480 y 800px */
@media screen and (min-width:480px) {
    .logo {
        background-image: url(logo-mediano.jpg);
    }
}

/* Logo grande, para anchos de pantalla de entre 800 y 1280px */
@media screen and (min-width:800px) {
    .logo {
        background-image: url(logo-grande.jpg);
    }
}

/* Logo gigante, para anchos de pantalla de más de 1280px */
@media screen and (min-width:1280px) {
    .logo {
        background-image: url(logo-gigante.jpg);
    }
}
```

En el caso de etiquetas IMG, haremos flexible la imagen dentro del rango mínimo y máximo de un layout:

```
img, video, object { max-width:100%; }
```

Eso hará que dentro del rango de ancho del elemento que envuelva a la foto (columna) la imagen se estire desde un mínimo y hasta un máximo. Como el máximo es su tamaño real, 100%, consideremos ese ancho al definir el ancho de su elemento contenedor, o viceversa: creemos la imagen al tamaño máximo al que llegará su elemento contenedor.

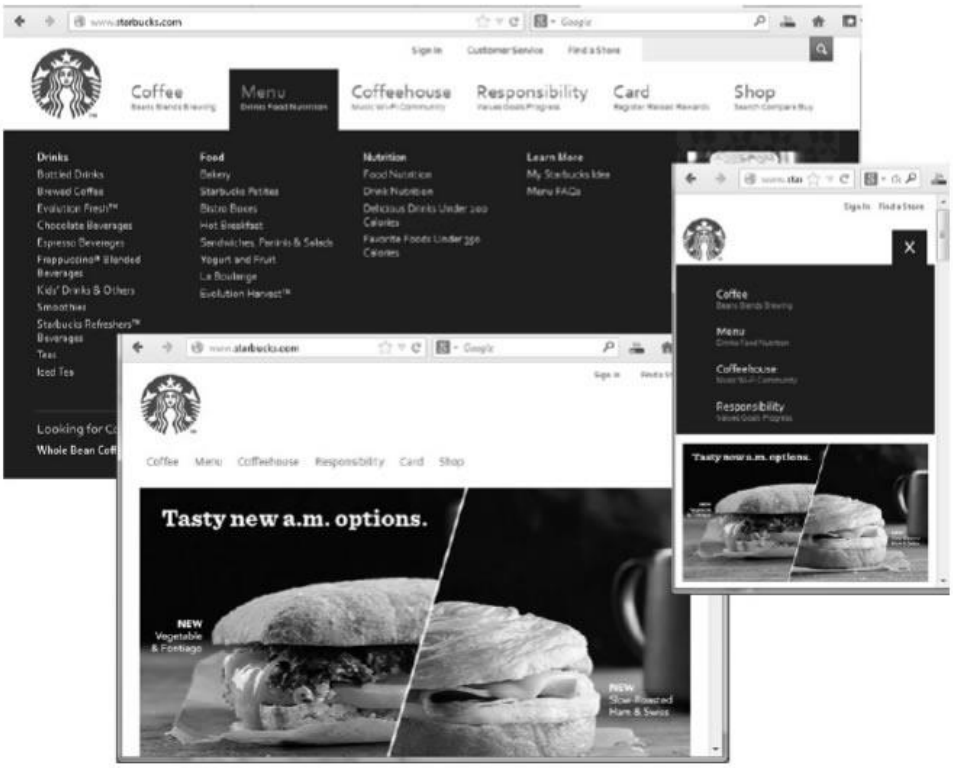
Existen varias propuestas del W3C para especificar que el navegador descargue distintas imágenes según el tamaño de pantalla (tal como en las imágenes de fondo) como por ejemplo, la posible etiqueta “picture” con varias etiquetas “source”, cada una condicionada por una media query, o que se modifique la etiqueta “img” permitiendo especificar más de un source mediante el atributo srcset. Al momento de escribir este libro ninguna de ellas está estandarizada ni implementada por los navegadores.

Otros casos son los de utilizar dibujos vectoriales escalables SVG en vez de una etiqueta IMG cuando lo que muestre la imagen sea un dibujo con pocos colores (logotipos, iconos, etc.).

4. Navegación adaptable

La navegación es otro de los puntos críticos que debe adaptarse en un sitio para permitir su uso en teléfonos y tabletas.

Tomemos como ejemplo la navegación de este sitio:



En el caso del PC, vemos una serie de submenús desplegables, a partir de una lista (ul) de primer nivel que contiene el nombre de cada sección en letras grandes, y con una segunda línea descriptiva.

Vemos cómo, en un tamaño mediano de pantalla (tablet), esa segunda línea desaparece, y el tamaño de fuente es menor. Ese cambio puede lograrse con algo tan sencillo como cambiar el font-size (ya lo vimos en el primer punto de este capítulo) y ocultar con display:none los subtítulos por defecto, haciéndolos visibles con display:block a partir de cierto tamaño de pantalla.

Por ejemplo:

```
.subtitulos {
    display:none;
}

/* Hacemos visibles los subtítulos para anchos de pantalla mayores a 1024px */
@media screen and (min-width:1024px) {
    subtitulos {
        display:block;
    }
}
```

Y en el caso de la navegación en la pantalla más pequeña (teléfono), vemos que se aplicó el mismo criterio pero para ocultar el enorme listado de submenús, dejando solamente los botones de primer nivel, haciendo que no floten uno al lado del otro y dándoles un tamaño grande, para que puedan ser fácilmente pulsados con los dedos en una pantalla táctil.

Otro caso bastante común es el de reemplazar un menú visualmente muy amplio que es el que se usará en la PC, por un select con options en el teléfono.

En el código HTML, que es el mismo para todos los dispositivos, estará el código HTML de ambos menús (el formulario con el select y la “ul” con botones).

Por ejemplo:

```
<ul id="menupc">
    <li>HOME</li>
    <li>CONTACTO</li>
</ul>

<form id="menumovil">
    <select>...etc... </select>
</form>
```

Desde la hoja de estilos, en la zona inicial de la hoja sin condiciones, ocultamos el menú de PC y mostramos el de móvil:

```
#menupc {
    display:none;
}
```

```
#menumovil {
    display:block;
}
```

Y en una media query, invertiremos esto para cuando estemos en pantallas grandes:

```
@media screen and (min-width:640px) {
    #menupc {
        display:block;
        /* Mostramos el menú de PC */
    }
    #menumovil {
        display:none;
        /* Ocultamos el select para teléfono */
    }
}
```

De esta manera, ya podemos manipular la visualización de diferentes herramientas de navegación, gracias a las media queries.

Configurando el Viewport

El tamaño de la “ventana” del navegador en un PC, siempre coincide o es menor que el tamaño de la pantalla. Es decir, o usamos el navegador maximizado, a pantalla completa, o lo achicamos un poco. Pero nunca es más grande que la pantalla.

En móviles, en cambio, o la ventana del navegador coincide con el tamaño de pantalla (siempre maximizada), o es mayor que el tamaño de pantalla, viendo solo una parte de ella por vez. Nunca es menor ya que no podemos “achicar” la ventana para que ocupe menos de una pantalla. Pero sí puede suceder lo contrario, que el contenido supere el tamaño de la pantalla porque estemos viendo solo una parte, debido a la utilización del zoom.

Por ejemplo: Safari y Opera Mini asignan 980px de ancho al viewport y hacen zoom para mostrar lo que suponen una web hecha para PC (y en el 99% de los casos, ¡aciertan!).

Veamos un ejemplo comparando la misma página sin que el navegador le haga zoom, “miniaturizándola”, y con la etiqueta viewport que lo impide, que veremos a continuación:

¿Cómo podemos impedir que los navegadores móviles escalen nuestros sitios y los miniaturicen? La solución es una nueva etiqueta meta cuyo name es **“viewport”**, que solo es aplicada por dispositivos móviles y es ignorada en computadoras de escritorio.

Su sintaxis es la siguiente:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

Dentro del atributo content, lo que estamos haciendo es definir que el width de la ventana del navegador tenga un valor “real”; es decir, que el navegador no nos “mienta”, sino que utilice como valor de su propiedad “width” el valor de otra propiedad, que es “device-width”, es decir, el ancho físico de su pantalla.

De esa manera, un navegador dentro de un dispositivo de por ejemplo, 320px de ancho, ya no declarará un width ficticio de 980px, sino que lo dejará en 320px, que es el ancho del dispositivo, con lo cual no miniaturizará nuestro diseño.

Por otro lado, notemos que hemos definido un valor inicial para el zoom, mediante la propiedad initial-scale a un valor de “1” que equivale a 100%, es decir, el tamaño natural. De esta manera, cuando el navegador ingrese a nuestra página, no aplicará ningún nivel de zoom que previamente el usuario hubiera configurado.

> Atención con el zoom: no debemos desactivar por completo la posibilidad de realizar zoom por parte del usuario, ya que ésta es una atribución suya. Pensemos, por ejemplo, en un usuario miope, que necesita ampliar parte de nuestro sitio. Para eso, evitemos definir en la etiqueta viewport los valores de mínimum-scale y de maximum-scale, lamentablemente muy difundidos.

Soluciones para navegadores que no entienden Media Queries

Los móviles más antiguos, cuyos navegadores no entienden media queries, no tendrán problemas en mostrar un diseño acorde a su tamaño si ese diseño es el primero en la hoja de estilos y no está envuelto dentro de ninguna condición. Los navegadores móviles más modernos procesan media queries, así que tampoco son un problema.

Las tablets son dispositivos relativamente nuevos, creados a partir del año 2009, con lo cual todos sus navegadores soportan media queries.

El único problema de compatibilidad lo tendremos en PCs de escritorio. Los navegadores de PC más modernos tienen soporte para media queries, pero algunos navegadores antiguos como Explorer 8 todavía subsisten, y no interpretan las media queries.

Para solucionarlo, usaremos un script compatibilizador llamado CSS3-mediaqueries-js que puede descargarse en:

```
https://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js
```

Simplemente vinculamos nuestra página HTML a ese script con una etiqueta <script> y a partir de ese momento el Explorer 8 interpretará las media queries.

A manera de conclusión, vimos que es perfectamente posible crear experiencias de navegación por sitios web multidispositivo, es decir, que se puedan usar con un diseño y unas herramientas optimizadas para diferentes tamaños de pantalla, si aprovechamos la técnica de media queries para crear esquemas tipográficos adaptables, layouts y rejillas flexibles, elementos visuales como imágenes y videos líquidos, y herramientas de navegación optimizadas para el uso en pantallas táctiles.