Maquetación y animaciones Web mediante HTML5 y CSS3

Responsive Web Design (RWD)

Módulo: **Desarrollo de Interfaces Web**

CFGS Desarrollo de Aplicaciones Web

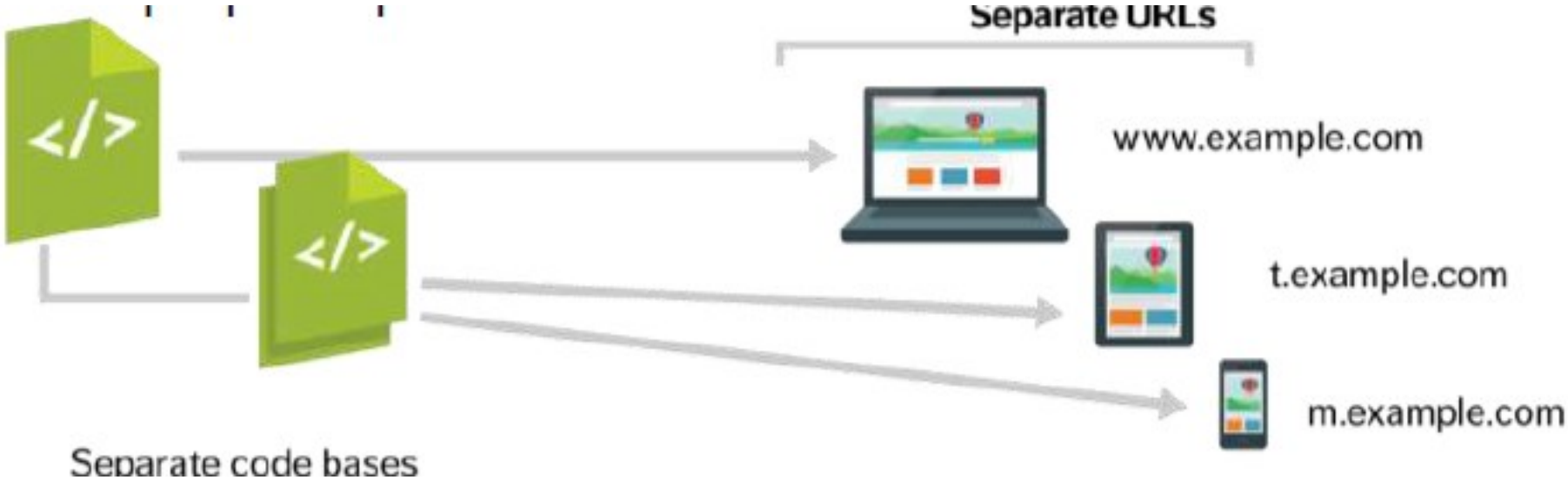
Introducción

Debido a que los diseñadores estaban acostumbrados a hacer páginas web con anchos fijos, la solución más fácil y más obvia era simplemente pasa por hacer sitios web para móviles (independientes) con un ancho de página fijo.

A los usuarios, si estaban usando un teléfono móvil, se les redirigía automáticamente a menudo a la versión móvil del sitio. En algunos casos, podrían elegir ir a la página web móvil haciendo clic en un enlace, o visitar el sitio a través de una URL diferente, generalmente mediante un subdominio m (por ejemplo, http://m.sprint.com).

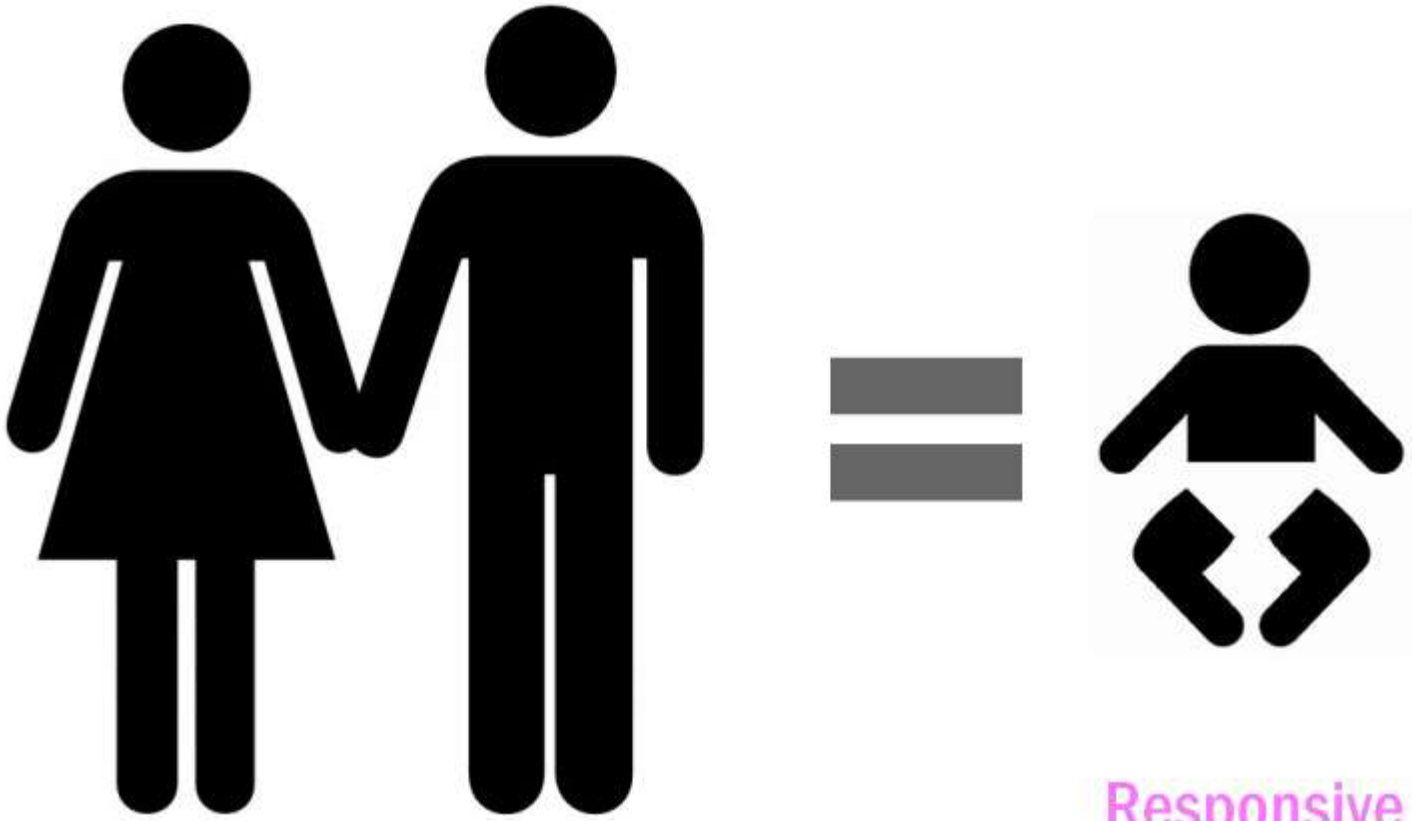
Introducción

Por supuesto, esto significa trabajo extra para el equipo de desarrollo web. Por lo general, el sitio móvil era una versión reducida de la página web regular con sólo una pequeña parte del contenido.



¿Qué es Responsive Web Design?

Nace en 2010



¿Qué es Responsive Web Design?

El término Responsive Web Design se citó por primera vez por Ethan Marcotte en su artículo A List Apart en Mayo del 2010.

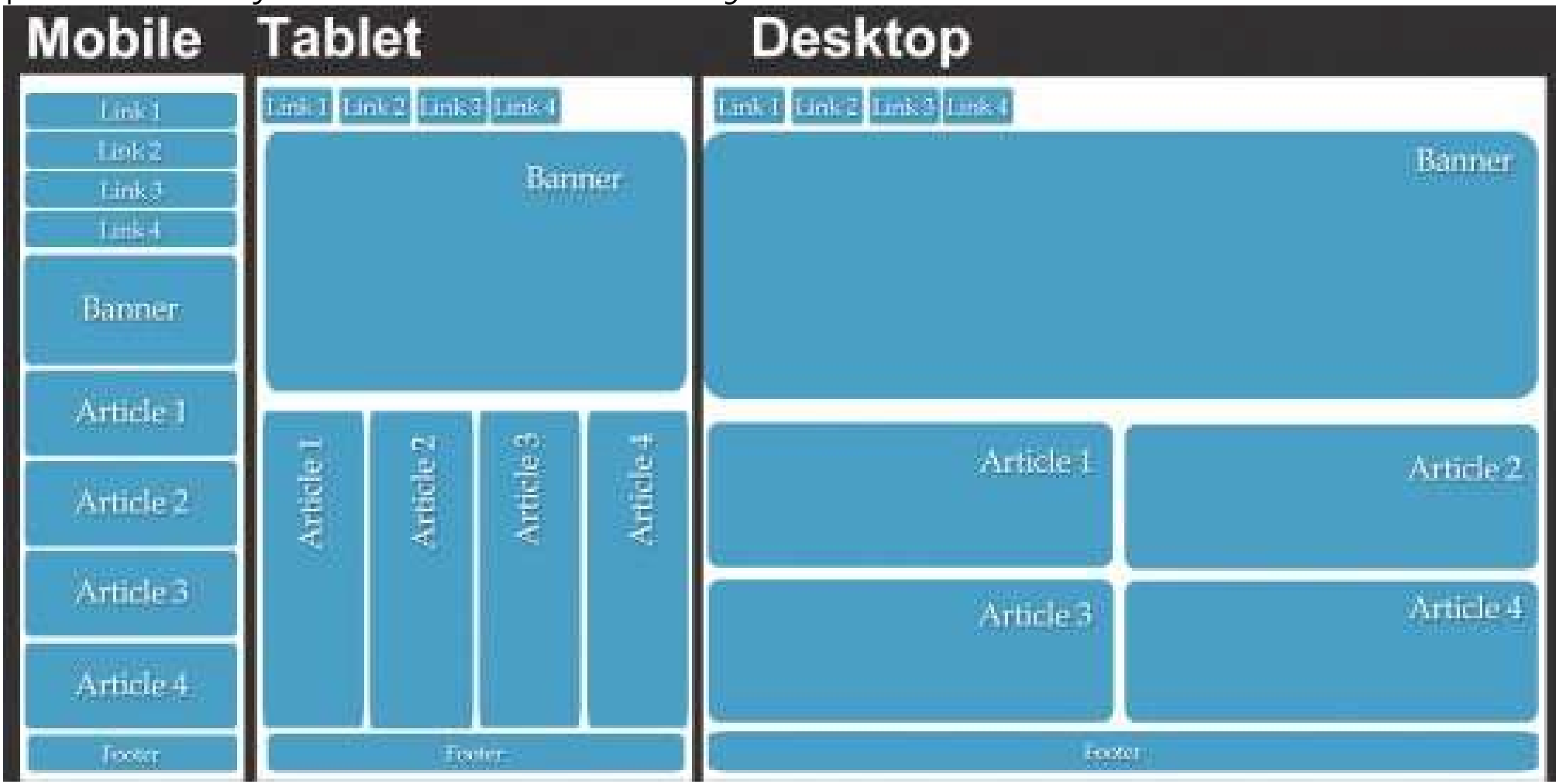
http://alistapart.com/article/responsive-web-design

Definió la técnica RWD utilizando fluid grids, imágenes flexibles, y media queries para ofrecer diferentes experiencias visuales para diferentes tamaños de pantallas.

Ethan extiende su teoría sobre RWD publicando el libro titulado Responsive Web Design.

¿Qué es Responsive Web Design?

Las cajas azules representan el código HTML. Las variaciones de posiciones y tamaños de esas cajas se controlan mediante CSS.



¿Qué es Responsive Web Design?

- Diseño cuya distribución de contenidos se adapta a los dispositivos del usuario en: tamaño de la pantalla, la plataforma y la orientación.
- El diseño y desarrollo deben responder al comportamiento del usuario.
- El sitio Web tiene la tecnología para responder automáticamente a las preferencias del usuario.

¿Diseño Responsive vs Diseño Adaptativo?

El **diseño responsivo** reestructura en la pantalla del dispositivo los elementos de la Web para optimizar todo el espacio disponible.

Por ejemplo, un menú horizontal se convierte en vertical si se visualiza en smartphones, evitando los scrolls infinitos y organizando el contenido de una mejor forma.

La única desventaja de esta técnica de diseño web es que hay elementos que no son redimensionables (como los vídeos y banners) que no cambian su tamaño dependiendo del dispositivo.

El **diseño adaptativo** posee distintos diseños para los distintos dispositivos móviles, uno para tablets y otro para smartphones, por ejemplo. Este tipo de diseño nos permite crear tantas versiones como dispositivos pensemos que se van a utilizar para acceder a nuestra web.

Webs SIN/CON Responsive Web Design

Webs SIN/CON Responsive Web Design

Ventajas de Responsive Web Design

- Se tiene que mantener un único documento HTML
- Se tiene que mantener un único documento CSS
- El sitio es accesible desde cualquier tipo de dispositivo
- Se mejora la UX (user experience). Lo que quiere decir, es que los usuarios tienen experiencias muy parecidas al acceder al sitio Web desde dispositivos diferentes
- Una Web responsiva es flexible y adaptable

¿Cuál es mejor?

Diseño Responsivo: Utiliza CSS3 media queries y rejillas basadas en proporciones. Todos los elementos continúan mostrándose por pantalla con cambio en los anchos.

Diseño Adaptivo: Utiliza CSS3 media queries para ocultar elementos en la pantalla, disminuyendo anchos.

Mejora Progresiva: Diseñar en primer lugar teniendo en cuenta la vista móvil, a partir de ahí, añadir elementos a la pantalla a medida que se incrementa el ancho.

Técnicas fundamentales

Hay 3 partes en el diseño Web responsivo:

1. Flexible, maquetación basada en rejillas

   Los sitios web se construyen utilizando unidades relativas (em o %) para los anchos en vez de px.

2. Media queries

   Dejan que la presentación del contenido se adapte a un rango específico de dispositivos de salida sin tener que cambiar el contenido en sí.

3. Media & imágenes flexibles

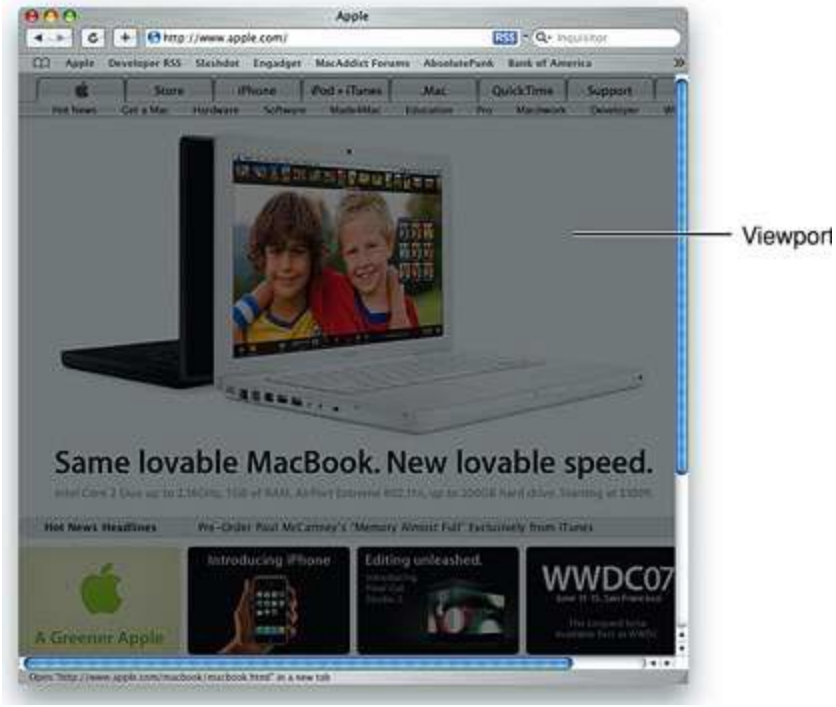
   Cuando el tamaño de la pantalla comienza a cambiar, las imágenes/media necesitan ser flexibles para adaptarse al tamaño de la pantalla.

Viewport

El **viewport** es el área de la pantalla (dentro de la ventana del navegador) en la que el sitio Web se muestra. Le dice al navegador como debe comportarse el navegador cuando se renderiza la página – le debes decir al navegador cómo de grande será el viewport.

*En un **equipo de escritorio**, si a la ventana del navegador les restas los menús, las barras de herramientas, las barras de desplazamiento, y todo lo demás que es parte del propio navegador, lo que queda en el interior es la ventana gráfica.*

*En un **dispositivo móvil,** el ancho de la ventana gráfica es el mismo que el ancho de la pantalla.*



Viewport

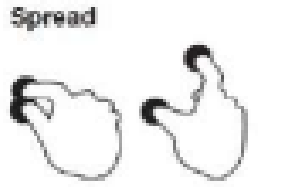

Viewport Fix
Without Meta Tag
With Meta Tag
AT&T 4G
4:23 PM
70%
Interface - Blog of Web Communications at the...
uablogs.missouri.edu/in
Google
Interface
Managing Slide Shows
PHP includes with Cascade Server
AT&T 4G
4:21 PM
70%
Interface - Blog of Web Communications at the...
uablogs.missouri.edu/in
Google
Interface
Blog of Web Communications at the University of Missouri
Managing Slide Shows
Published on February 29, 2012 by Josh Nichols | Posts by This Author

Viewport

Si vas a utilizar RWD, es bueno tener el tag viewport de la siguiente forma:

```
<meta name="viewport" content="width=device-width,initial-scale=1">
```

- **width=device-width** → "mi página se adapta a tu ancho"
- **initial-scale=1** →no hagas zoom-in



Ampliar o agrandar un objeto, Zoom in

Por ejemplo, un valor de initial-scale=2 podría significar que la página podría ser aumentada 2 veces el largo de su tamaño actual, así que se vería la mitad de una página en la pantalla.

Explicación detallada de la etiqueta viewport

http://www.paulund.co.uk/understanding-the-viewport-meta-tag

Viewport

Hay 2 formas de incluir la etiqueta viewport para modificar el comportamiento del viewport por defecto del user agent (del navegador).

1. Regla CSS @viewport. Relativamente nuevo.

```
/* Documento CSS */
@viewport {width: 480px; zoom: 1;}
```

2. viewport en etiqueta meta. Universalmente extendida.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Viewport

- width = ancho del área mostrada
- device-width = ancho del dispositivo
- orientation = orientación del dispositivo. Posibles valores: landscape y portrait. Ejemplo:

  media="screen and (orientation:portrait)“
- aspect-ratio = indica la proporción width/height expresándola mediante 2 números separados por “/”. Por ejemplo:

  media="screen and (aspect-ratio:16/9)“
- resolution = densidad de pixeles del dispositivo (dpi). Por ejemplo:

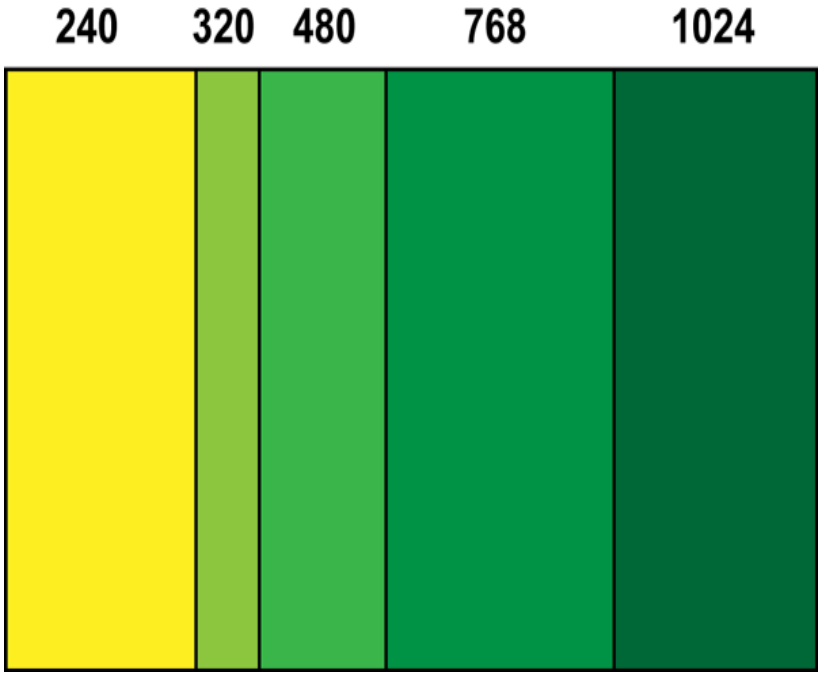
  media="print and (resolution:300dpi)"

Responsive Web Design - Anchos

Los usuarios pueden llegar a una web a través de una gran variedad de dispositivos y cada uno con una resolución de pantalla distinta. Desde los 240 píxeles de ancho de algunos smartphones hasta los más de 1200 que puede tener un monitor, hay muchas variantes, pues, para complicar aún más la situación, en móviles y tabletas las páginas pueden verse en horizontal o en vertical.

Las resoluciones de dispositivos más comunes pueden ser divididas en 4, las cuales pueden ser tu punto de partida:

- Menores a 480 px que serán los smartphones de primera generación.
- Menores de 768 px para teléfonos de última generación y algunas tablets.
- Mayores a 768 px que comprenderá cualquier computadora de escritorio, laptops y tablets.
- Mayores a 1024 px monitores de más de 13 pulgadas, pantallas y iPads que utilicen retina display.



Responsive Web Design – Media Queries

Dependiendo de las características del dispositivo donde se consulte el sitio.

Por ejemplo, una consulta que responde con un ancho máximo de 450 píxeles estaría dirigida a los navegadores móviles solamente, por lo que el CSS haría referencia a la siguiente condición:

```
<link href="css/phone.css" rel="stylesheet" type="text/css"
media="only screen and (max-width: 450px)">
```

Responsive Web Design – Media Queries

Formas de incluir media queries

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

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

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

Responsive Web Design – Media Queries

Las media queries son proposiciones lógicas. 'Si' la condición dada se cumple en el navegador, se ejecuta esa porción de código css.

CHULETA

- min-width: si la ventana es mayor que...
- max-width: si la ventana es menor que...

and

```
@media (min-width: 600px) and (max-width: 800px) {
      html { background: red; }
}
```

or

```
@media (max-width: 600px) ,(min-width: 800px) {
      html { background: red; }
}
```

Responsive Web Design – Media Queries

El operador not no puede ser usado para negar un query individual, solo para un query completo. Por ejemplo, el not en el siguiente query es evaluado al final:

@media not all and (monochrome) {...}

- se evalúa: @media not (all and (monochrome)) {...}
- y no: @media (not all) and (monochrome) {...}

Un not negará un query si es posible pero no a todos los query posibles si están ubicados en una lista separada por comas.

@media not screen and (color), print and (color) {...}

- se evalúa: @media (not (screen and (color))), print and (color) {...}

Responsive Web Design – Media Queries

Exclusivas

```
@media (max-width: 400px) {
   html { background: red; }
}
@media (min-width: 401px) and (max-width: 800px) {
   html { background: green; }
}
@media (min-width: 801px) {
   html { background: blue; }
}
```

Sobreescribir

```
@media (min-width: 400px) {
   html { background: red; }
}
@media (min-width: 600px) {
   html { background: green; }
}
@media (min-width: 800px) {
   html { background: blue; }
}
```

Responsive Web Design – Media Queries

Mobile First

```
html { background: red; }
@media (min-width: 600px) {
  html { background: green; }
}
```

Desktop Firts

```
html { background: red; }
@media (max-width: 600px) {
  html { background: green; }
}
```

Responsive Web Design – Media Queries

@media
only screen and (-webkit-min-device-pixel-ratio : 1.5),
only screen and (min-device-pixel-ratio : 1.5)

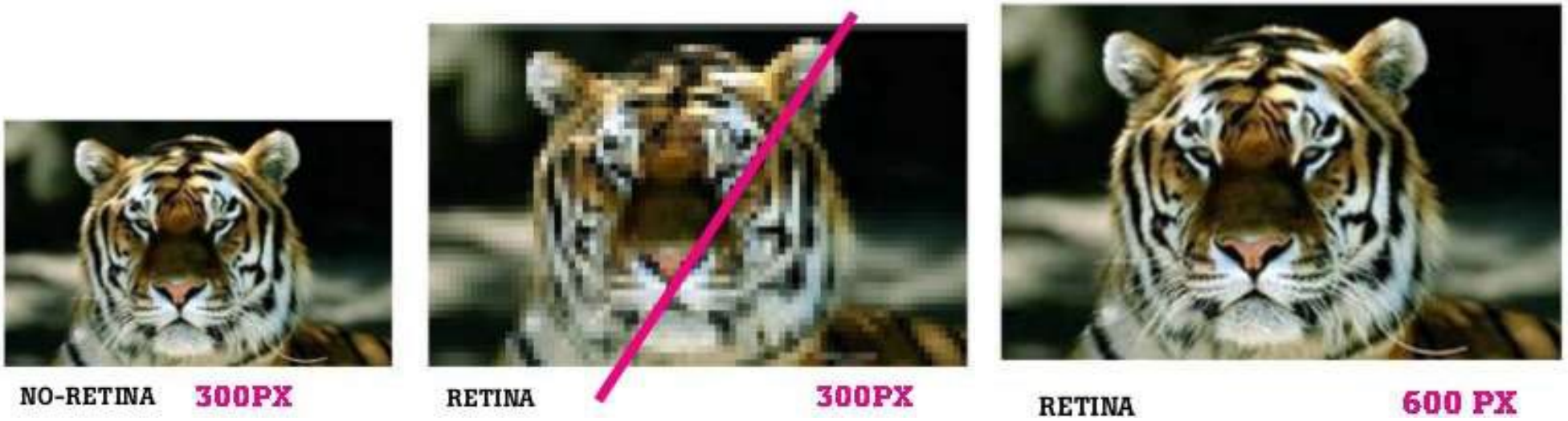
```
{
   body {
      font-size: 90%;
   }
}
```

- only es un operador lógico
- No lo reconocen user agents que se basan en HTML4
- Oculta los estilos a dispositivos o navegadores que no soportan las media queries

Responsive Web Design – Retina Display

Con la aparición de dispositivos con alta densidad de píxeles en pantalla (DPI) lo que deberemos de hacer es guardar las imágenes con una mayor de densidad de píxeles (ocupando más espacio).

En general las imágenes se suelen almacener con una densidad de 96 DPI. Para retina display se suelen guardar con 300 DPI (aprox. el doble).



Responsive Web Design – Fuentes

En el RWD el tratamiento de los textos no está definitivamente fijado y resuelto. Una buena estrategia puede ser utilizar media queries y las unidades em:

```
#contenedor_principal {
    font-size: 16px
}

p {
    font-size:0.875em
}

h1, h2, h3 {
    font-size:1.5em;
}

#bloque p {
    font-size:1.2em;

}
```

En el contenedor principal → fijar el tamaño de la fuente en **px**
Crear reglas para los elementos que están dentro del contenedor principal en **em**

Como regla general, utiliza el font-size de mayor tamaño que no parezca desproporcionada y que en cada línea tenga 20 palabras o menos (cuentan todas las palabras).

Responsive Web Design – Imágenes, objetos, vídeos, etc.

Uno de los problemas más importantes es el de las imágenes y videos. ¿Cómo adaptar el tamaño de las imágenes en función de la resolución del dispositivo? Para ello utilizaremos CSS y las propiedades **width, max-width, min-width** y las medidas se darán en %. Cuando indicamos max-width=100% forzamos a que siempre ocupe el ancho del contenedor pero no supere el tamaño original.

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

object {
    max-width: 85%;
}

video {
    max-width: 50%;
}
```

Responsive Web Design – Imágenes, objetos, vídeos, etc.

Podemos tener imágenes adaptadas a cada uno de los breackpoints de las media queries. Les asignaremos clases diferentes:

```
<img class="min" src="./CanariasMin.png" width="60" height="70">
<img class="mid" src="./CanariasMid.png" width="80" height="93">
<img class="max" src="./Canarias.png" width="100" height="116">
```

Ocultaremos las imágenes que no queremos mostrar en esa media querie y mostraremos la que sí:

```
@media screen and (min-width:801px) and (max-width:10000px) {
.min, .mid {display:none;}
.max {display:block}
}
```

```
@media screen and (min-width:400px) and (max-width:800px) {
.min, .max {display:none;}
.mid{display:block}
}
```

Responsive Web Design – Sugerencia

Siempre separar el CSS de la estructura del CSS para el formato:

- **Formato** → color, background-color, font-size
- **Estructura** → width, height, padding, margin

De esta manera es más fácil establecer las diferencias entre las media queries.

Responsive Web Design – Diseño

Existen **dos enfoques** para la etapa de diseño:

- Centrado en dispositivos: Se hacía una lista de 5 o 6 dispositivos (los más vendidos), se anotaba el ancho en pixeles de cada dispositivo y se definían los breakpoints con una media querie en pixeles para cada resolución. ¡¡¡Insostenible!!!

- Centrado en contenidos: Se evaluán los anchos de las filas del contenido, se hace una lista de anchos de ventana del navegador a los que ese contenido ya no es legible y se hacen los breakpoints en función de esos anchos (en em).

```
@media screen and (max-width: 26em){ /* Primer Diseño */}
@media screen and (min-width: 26.01em) and (max-width: 37em)
{/* Segundo diseño */ }
@media screen and (min-width:37.01em){ /* Tercer diseño */}
```

¡¡¡Sostenible!!!

Responsive Web Design – Centrado en contenidos

1. Inventario de Contenidos (listado total).

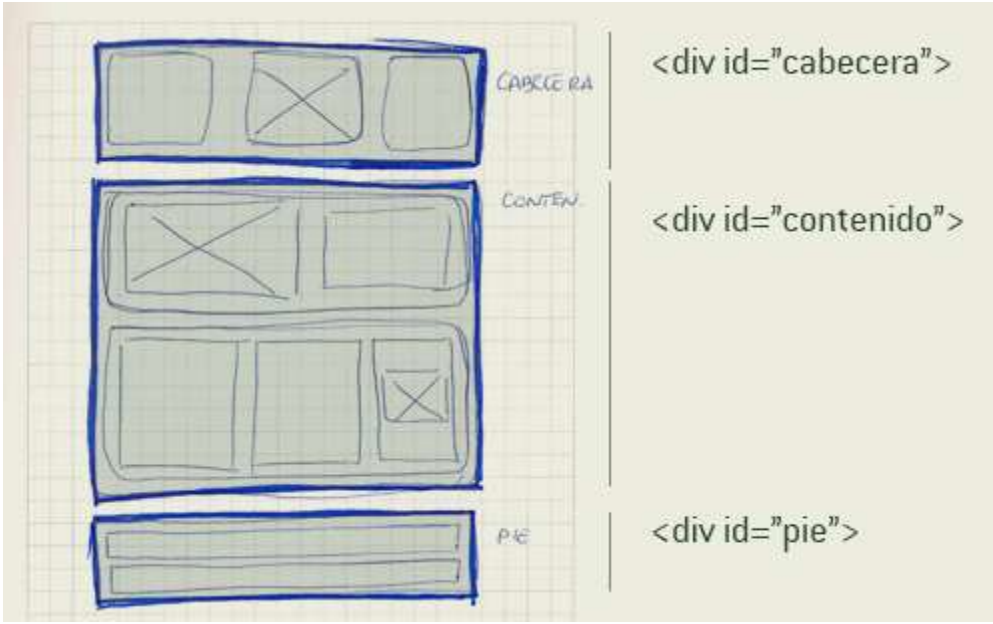
2. Mapa del sitio.

3. Definición de tipos de Plantillas (para portadas, secciones, fichas de contenido, tipos de recursos multimediales).

Responsive Web Design – Centrado en contenidos

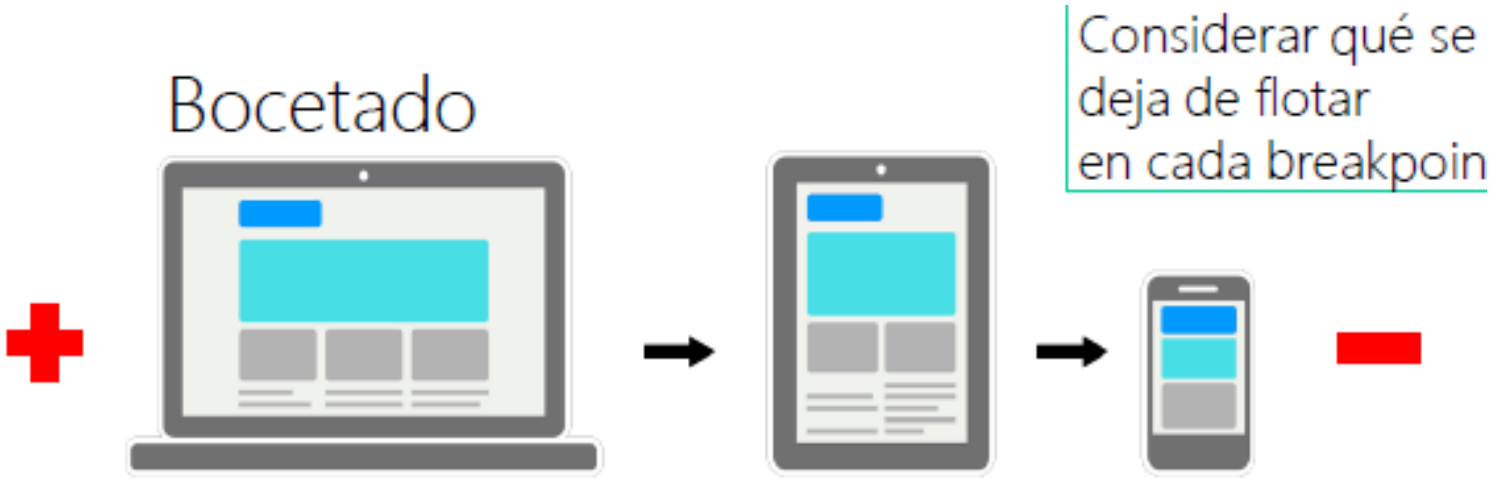
Primero priorizamos contenidos

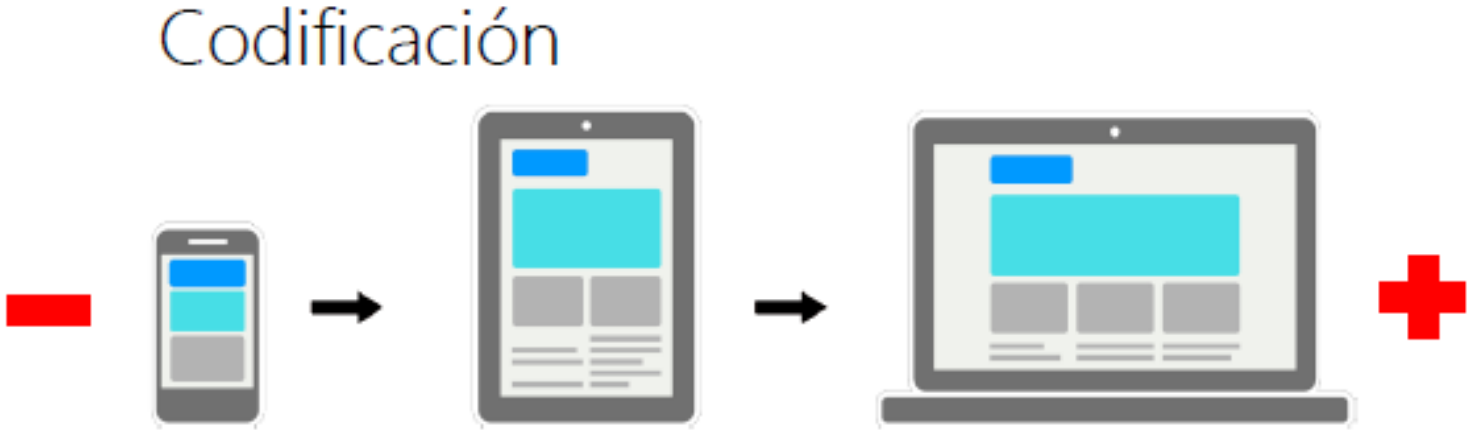
1.
2.
3.



Responsive Web Design – Centrado en contenidos

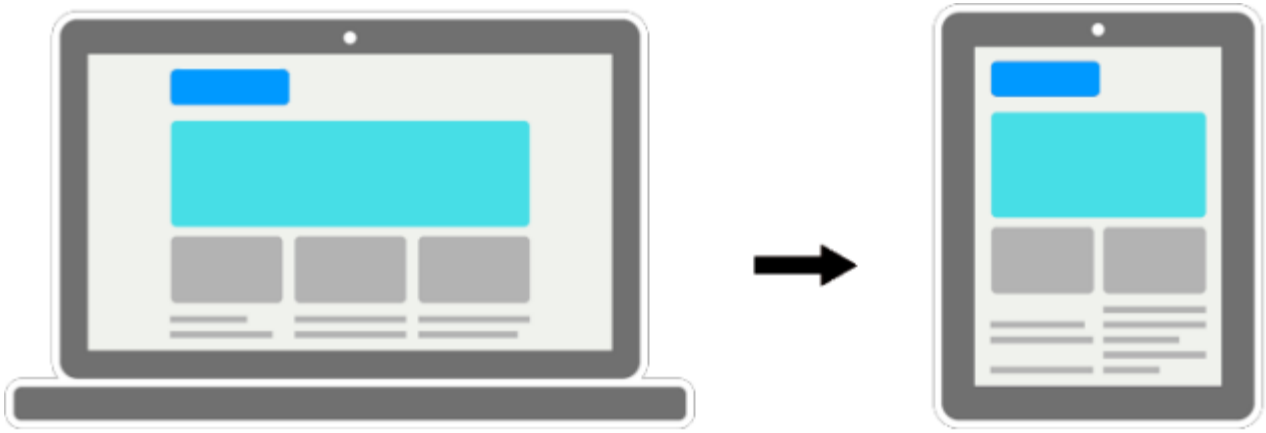
Con los contenidos priorizados para una plantilla, bocetamos de mayor a menor y codificamos de menor a mayor.





Responsive Web Design – Centrado en contenidos

1. Mantener igual (logo azul)
2. Dejar que se ajuste ancho en porcentaje (slider celeste)
3. Dejar de flotar y aumentar ancho en porcentaje (columnas grises)
4. Ocultar
5. Mostrar

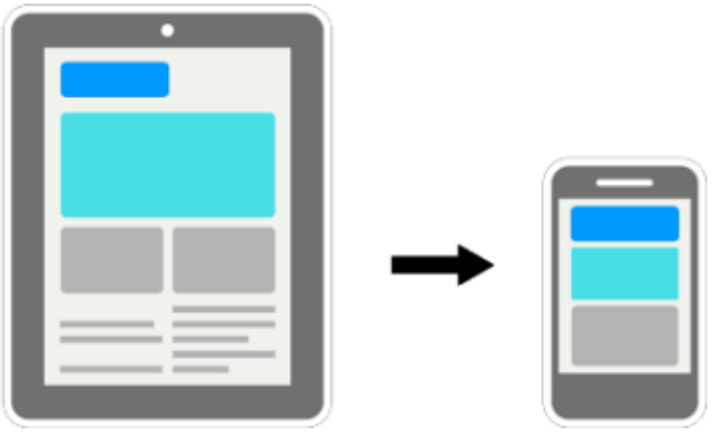


Los **breakpoint se definen en función del contenido**. Es decir, el contenido decide, y tendremos que estirar la ventana hasta romper el diseño y medir usando las reglas de las herramientas de desarrolladores u otro plugin.

Responsive Web Design – Centrado en contenidos

Repetir eso “n” veces…

1. Mantener igual (logo azul)
2. Dejar que se ajuste ancho en porcentaje (slider celeste)
3. Dejar de flotar y aumentar ancho en porcentaje (columnas grises)

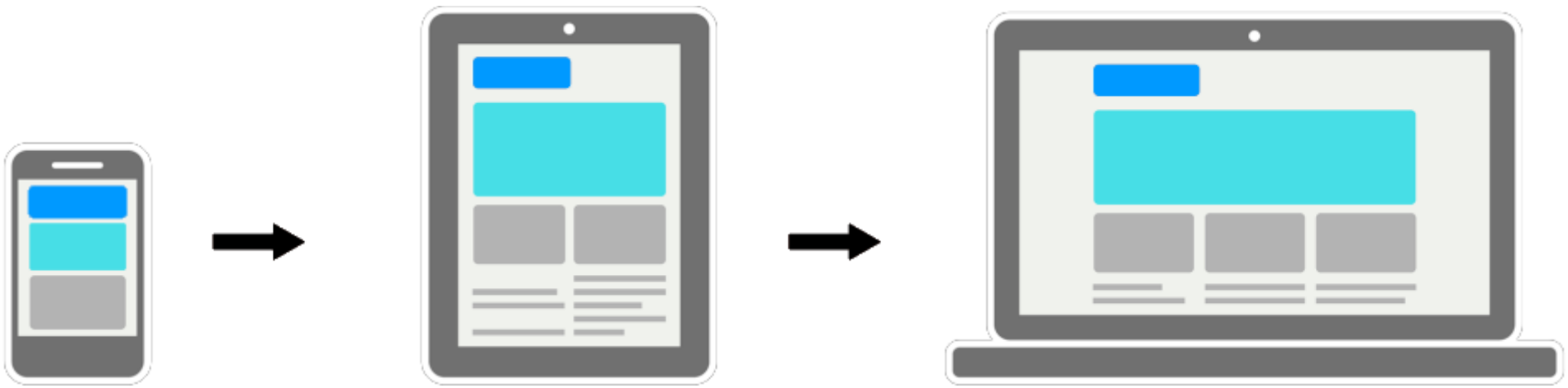


Hay que probar cada boceto hasta los extremos

Responsive Web Design – Centrado en contenidos

En los primeros años del desarrollo responsive (hasta 2012) el proceso de diseño ha sido el de "adaptación" de la versión de escritorio a móvil con la técnica "Graceful Degradation".

Sin embargo desde 2012 se emplea un enfoque "Mobile First" en lo que se opera a la inversa. La experiencia de usuario debe de ser la misma desde la version móvil.



Solo HTML **Flotar, estirar (CSS)** **Flotar, estirar (CSS)**

Responsive Web Design – Herramientas

Es IMPOSIBLE realizar el testeo de una aplicación web móvil dado que no es posible tener todos los dispositivos reales para realizar pruebas.

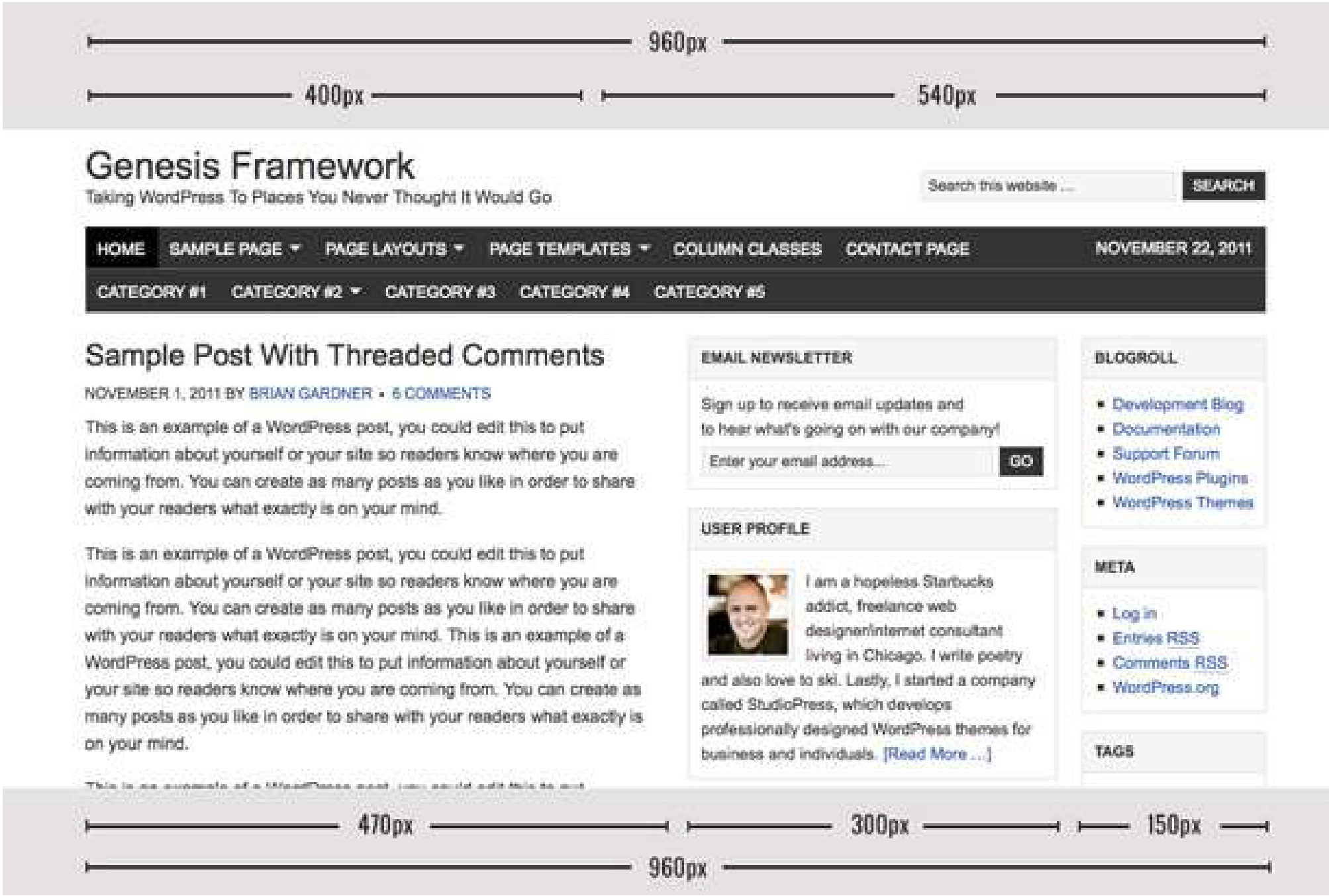
Extensiones Chrome:

- Web Developer
- Responsive Design Testing
- Responsive Site View

Herramientas Online (Existen webs más o menos fiables para realizar pruebas):

- Screenfly
- Responsinator

Responsive Web Design – Ejemplo paso a paso



Responsive Web Design – Ejemplo paso a paso

```
<div id="wrap">
   <div id="header">
      <div id="title-area"></div>
      <div class="widget-area"></div>
   </div>
   <div id="inner">
      <div id="content-sidebar-wrap">
         <div id="content"></div>
         <div id="sidebar"></div>
      </div>
      <div id="sidebar-alt"></div>
   </div>
</div>
```

Responsive Web Design – Ejemplo paso a paso

```
#wrap { width: 960px; }
#header { width: 960px;}
#title-area { width: 400px;}
#header .widget-area { width: 540px;}
#inner { width: 960px;}
#content-sidebar-wrap { width: 790px;}
#content { width: 470px;}
#sidebar { width: 300px;}
#sidebar-alt { width: 150px;}
```

Responsive Web Design – Ejemplo paso a paso

```
#wrap { width: 100%; }
#header { width: 100%;}
#title-area { width: 41.666667%;}
#header .widget-area { width: 56.25%;}
#inner { width: 100%;}
#content-sidebar-wrap { width: 82.291667%;}
#content { width: 48.958333%;}
#sidebar { width: 31.25%;}
#sidebar-alt { width: 15.625%;}
```

41.666667% = (400px/960px)*100

Responsive Web Design – Ejemplo paso a paso

```
.widget-area ul {
   margin: 10px 0 0 25px;
}
```

Transformamos px a %

```
.widget-area ul {
   margin: 10px 0 0 16.666667%;
}
```

El ancho de la barra lateral es de 150px (`#sidebar-alt`).

El margin-left de 25px convertido a % sería: (25/150) * 100 = 16.666667

Responsive Web Design – Ejemplo paso a paso

- Abre el fichero mediaqueries.css (carpeta rwd) que tiene algunos ejemplos de media queries.
- Abre el fichero rwd.html de la carpeta rwd.

Primeramente vamos a convertir el CSS para que sea fluido con el cálculo target/context considerando un ancho de navegador de 1680 pixels.

Al convertir a porcentaje el width de #blog tenemos

1020 px / 1680 px = 0.60714285714285714285714285714286 =

60.714285714285714285714285714286%

Responsive Web Design – Ejemplo paso a paso

Trabajaremos con **rwdconv1.html**

- Insertaremos el siguiente media query después de #blog p

```
@media screen and (max-width:480px) {
   #blog {
      float: none;
      width: 92.43137254901960784313725490196%;
      background-color:#FFB3B3;
   }
}
```

- Cambiamos el background color para resaltar el cambio que se produce al utilizar la media query.

Responsive Web Design – Ejemplo paso a paso

Prueba la página con la nueva media query

- Modifica el tamaño del navegador
- Observa que algo no va bien con la etiqueta *article*

➔ ¿Qué podemos hacer?

Nota: utiliza **rwdconv2.html** (con el media query implementado)

Responsive Web Design – Ejemplo paso a paso

Añade el siguiente query para los elementos `aside` y `article` después del último media query que añadimos:

```
@media screen and (max-width:830px) {
#blog .aside{
   float: left;
   width: 98%;
   background-color:#95C9E8;
   margin-top:5px;
}
#blog .article{
   float: left;
   width: 98%;
   background-color:#B0E6C6;
   margin-top:10px;
   }
}
```

Responsive Web Design – Ejemplo paso a paso

A continuación, de **rwdconv3.html** observa:

- El float para article y aside
- El width

Último "ajuste" - aumentar el ancho de la sección blog para adecuarnos al tamaño pequeño de los dispositivos móviles.

Puedes utilizar **rwdconv4.html** y observa que el width está al 95%.

Responsive Web Design – Ejemplo paso a paso

En el pasado se definía en pixeles el ancho y alto de nuestras imágenes y esto no es flexible.

Observa **rwdimg1.html** en la carpeta **rwd,** encontraras siguiente el código entre los párrafos primero y segundo del artículo:

```
<img src="servers.gif" alt="Servers" title="Servers">
```

El truco para convertir una imagen en flexible es muy simple:

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

Añade esto a **rwdimg1.html** y observa la diferencia cuando cambias el tamaño del navegador.