Capítulo 6 Responsive WEB

Responsive Web Design (o diseño web adaptativo) se trata de una técnica de diseño y desarrollo web por el que se consigue que un **único sitio se adapte perfectamente a todos los dispositivos** que puedan consumirlo, desde ordenadores de escritorio a netbooks, tablets, teléfonos móviles, televisores, etc. En definitiva, se trata de construir una única web para que se vea correctamente y aproveche las particularidades de todo dispositivo que hoy exista, o pueda existir en el futuro, independientemente de la pantalla en la que se muestre.

6.1 Ventajas

Las ventajas de utilizar un diseño web adaptativo son más que evidentes:

• Con **una sola versión** en HTML y CSS se cubren todas las resoluciones de pantalla, es decir, el sitio web creado estará optimizado para todo tipo de dispositivos: PCs, tabletas, teléfonos móviles, etc. Esto mejora la experiencia de usuario a diferencia de lo que ocurre, por ejemplo, con sitios web de ancho fijo cuando se acceden desde dispositivos móviles.

• **Reducción de costos**. Se reducen los costes ya que hasta hoy se debe hacer un portal para la Web y otro para dispositivos móviles. Esto origina mayores costos de creación y mantenimiento de la información.

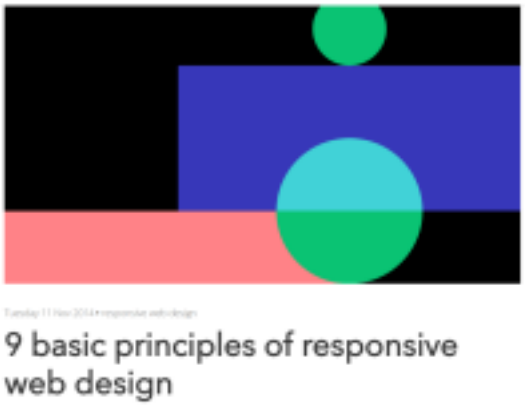
• **Eficiencia en la actualización**. El sitio solo se debe actualizar una vez y se ve reflejada en todas las plataformas.

• **Mejora el posicionamiento SEO.** Google tiene distintos tipos de robots que acceden y escanean los sitios web indexados en su buscador emulando a los diferentes dispositivos que navegan por internet. Estos robots, valoran la accesibilidad de la web en los diferentes dispositivos y la tienen muy en cuenta a

la hora de posicionar la web en los resultados de búsqueda.

Conceptos básicos

En el artículo 9 basic principles of responsive web design, de Froont, hay una explicación visual de algunos **conceptos básicos** necesarios para entender correctamente el Responsive Web Design, junto a unas animaciones muy representativas.

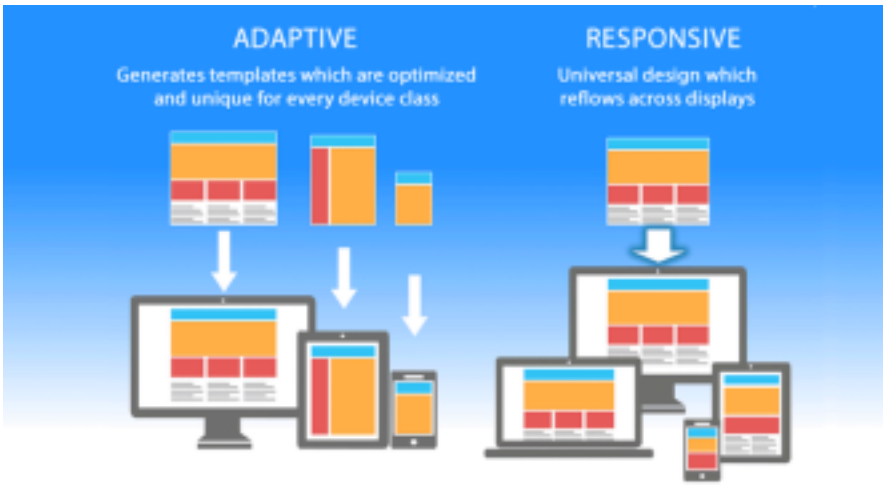


- **Conocer los formatos:** Monitores, smartphones, tablets u otros.
- **Conocer el público de tu sitio web:** ¿Acceden más usuarios desde móvil o desde escritorio? ¿Predominan las tablets o los móviles? ¿Tu objetivo es tener más usuarios de móvil o de escritorio?.

  Consulta con algún sistema de estadísticas como Google Analytics para comprobar que tipo de público tienes actualmente. También es aconsejable echar un vistazo a información externa como las que nos proporcionan estadísticas globales anónimas de Global StatCounter, para hacernos una idea de los atributos más comunes.

6.2 Diseño Web Responsivo vs. Diseño Adaptativo

La diferencia entre el diseño responsivo y el diseño adaptativo es que el diseño responsivo adapta la representación de una única versión de la página. Por el contrario, el diseño adaptativo entrega múltiples versiones completamente diferentes de la misma página.



El diseño web responsivo hace uso de media queries y CSS flexible que permiten modificar el layout y los elementos en función del tamaño de la pantalla o la resolución del dispositivo. No requiere versiones separadas para diferentes tamaños de pantalla; todo el contenido se adapta al tamaño de la ventana.

El diseño web adaptativo (Adaptive Web Design, AWD) utiliza una serie de diseños predefinidos que se seleccionan según el tipo de dispositivo o el tamaño de la pantalla. En lugar de redimensionar los elementos, se cargan diferentes versiones del diseño de la página según el dispositivo que se esté utilizando. Cada diseño se adapta de manera específica a un grupo de tamaños de pantalla predeterminados.

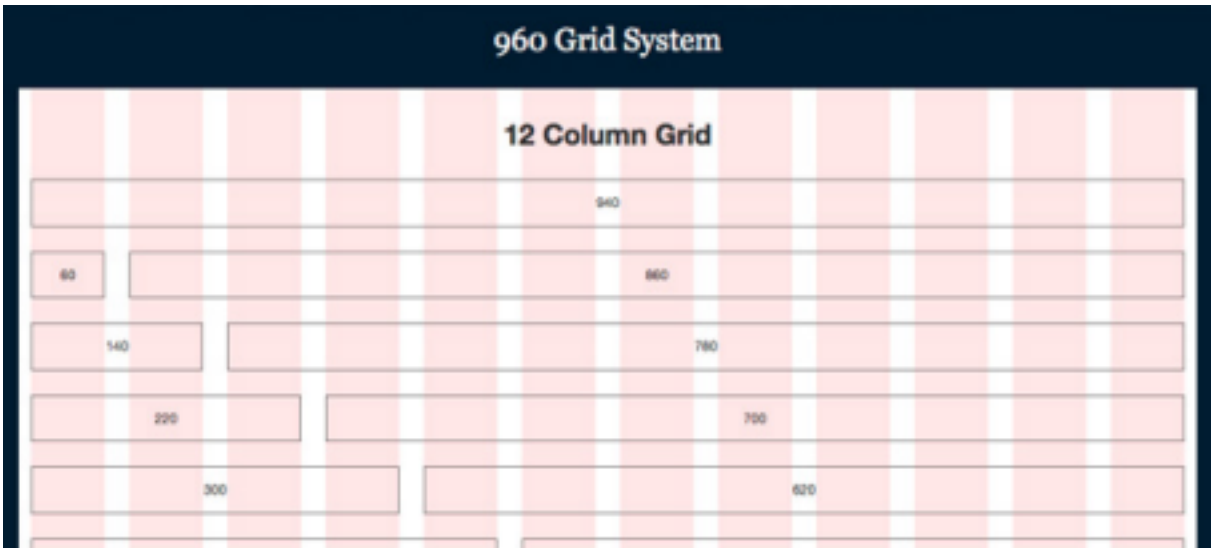
6.3 Los ingredientes Web Responsive

Los elementos básicos necesarios para construir un diseño adaptativo, son al menos los tres siguientes:

- Una estructura flexible, basada en rejilla.
- Imágenes, tablas, multimedia... flexible.
- *Media Queries*, como parte de la especificación de CSS 3.

6.3.1 Estructura de rejilla flexible

Los diseños basados en rejilla han sido utilizados ampliamente en sectores como el diseño gráfico, pero tardaron en ser adoptados por la web. Estos diseños consisten en una serie de columnas que permiten ordenar y estructurar los contenidos (texto, imágenes, media...) de una manera ordenada y uniforme. Una estructura clásica de rejilla es la que se muestra en la siguiente imagen:



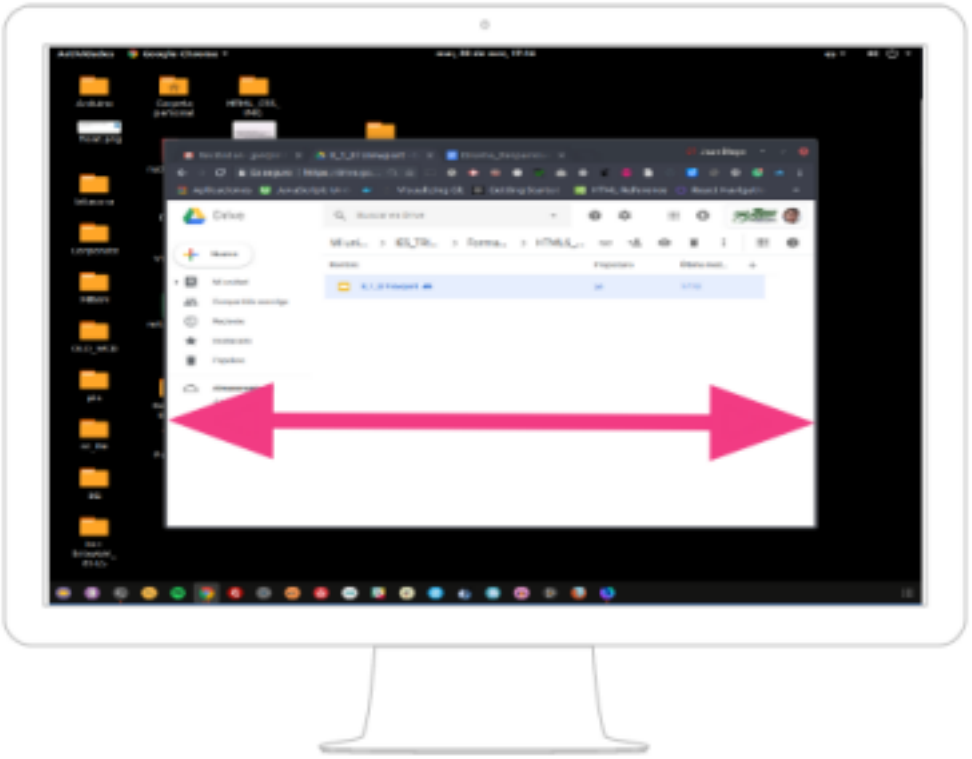
Se pueden observar claramente las líneas verticales y horizontales utilizadas para posicionar los elementos en el lienzo, así como los márgenes que separan los distintos elementos. De esta manera tan simple, es posible crear diseños con un aspecto visual uniforme.

6.3.2 VIEWPORT

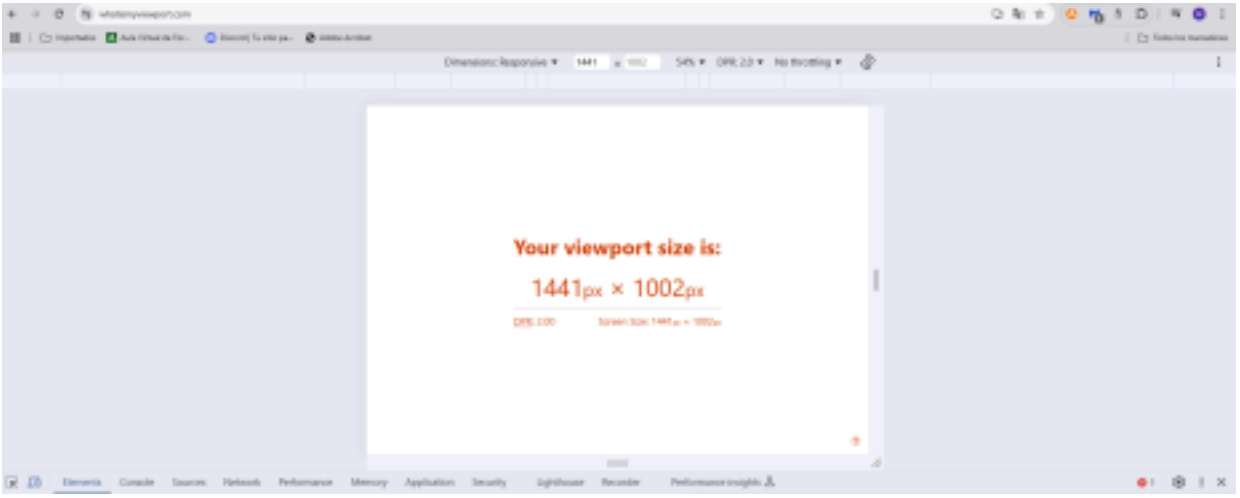
Uno de los conceptos más importantes que debemos conocer antes de lanzarnos a realizar diseños responsivos es el concepto de **Viewport**.

> "*Área de la pantalla en la que el navegador puede renderizar contenido, es decir, el espacio disponible para mostrar mi página web.*"

Este concepto es muy fácil de comprender cuando nos referimos a él en sistemas de escritorios o en laptops o portátiles. En estos casos el Viewport coincide con la pantalla de nuestro navegador.



Ej: https://whatismyviewport.com/



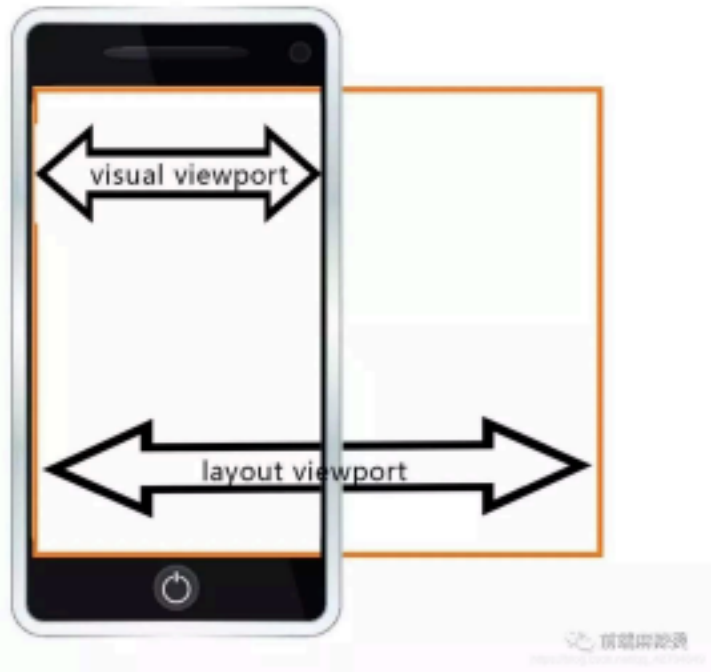
Sin embargo, si nos referimos a móviles y tablets estaremos hablando de otra cosa diferente.

Cuando este tipo de dispositivos irrumpieron, las páginas se maquetan usando normalmente una anchura fija en píxeles que solía variar entre 800px y 1000px que, obviamente, era mayor que la anchura de la pantalla de los móviles.

Ante esta circunstancia los fabricantes hicieron que este tipo de dispositivos tuvieran dos Viewports:

- El *Layout-viewport* que es el viewport que es tenido en cuenta para la aplicación de estilos. Suele ser de aproximadamente 960 px.

- El *Visual-viewport* que es el viewport que realmente ve el usuario.



Además, sucede que en este tipo de dispositivos podemos hacer Zoom mediante gestos lo que provoca que:

- No cambie el *Layout-viewport*
- Cambie el *Visual-viewport*.

Y no acaban ahí los problemas. También, en este mundo de miles de dispositivos diferentes no vamos a encontrar con que tenemos que lidiar con diferentes valores para los:

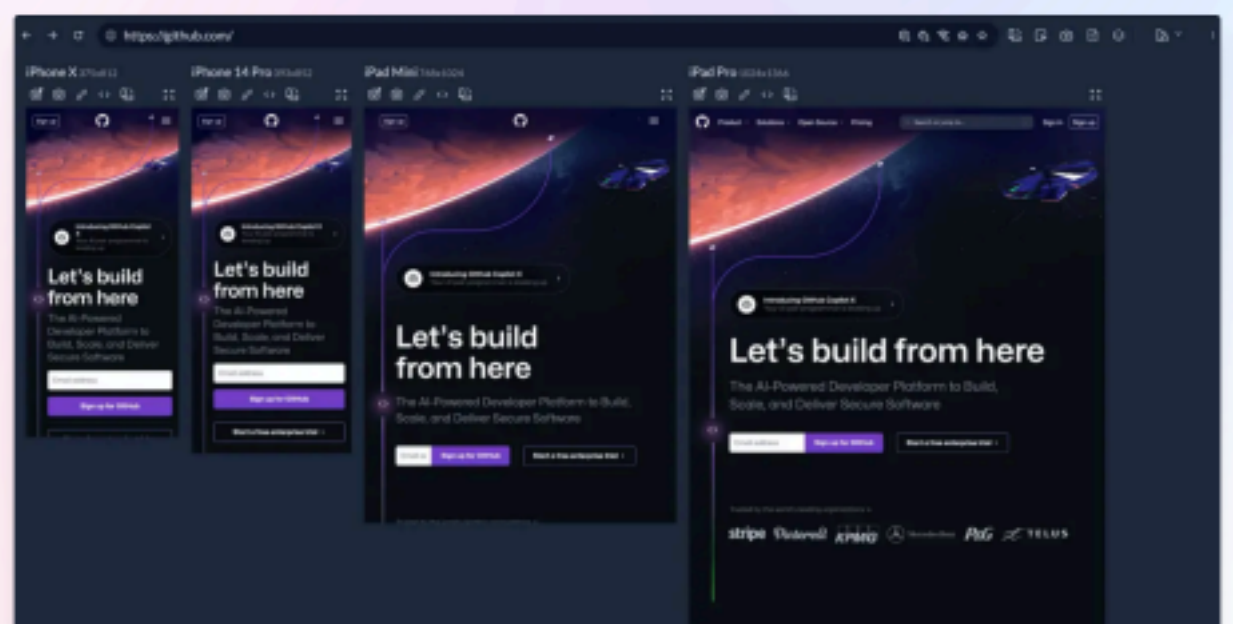
- Hardware Pixels: son los píxeles de resolución que nos da la tarjeta gráfica. ● DPI(Device Independent Pixels): Que es la unidad de medida del navegador. Se relaciona con la distancia real y ocupan lo mismo independientemente de la densidad de píxeles de la pantalla.
- DPR(Device Pixel Ratio): Hw Pixel / DPI (es una dimensión).

Para gestionar toda esta situación, se añadirá una línea en la cabecera de la página Web como esta:

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

Con esa simple línea podremos empezar a maquetar diseños responsivos de manera eficiente e incluso haremos que las páginas que no están preparadas para móviles se muestren correctamente aunque tengamos, eso sí, que hacer continuos zooms para poder usarlas.

App donde probar todas las versiones de tu web, https://responsively.app/:

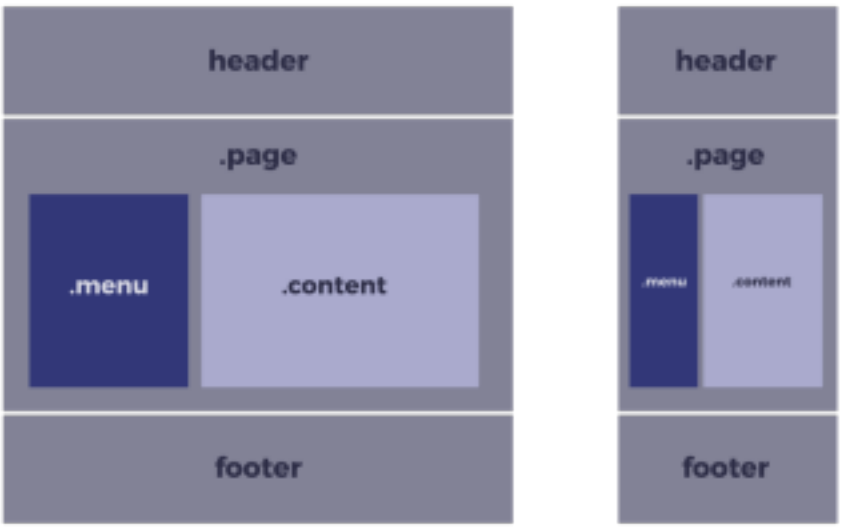


6.4 Crear una rejilla flexible

Mientras que los media query ofrecen la potencia real para desarrollar un sitio web

adaptativo, es posible ahorrar trabajo y código utilizando una aproximación en base a rejillas flexibles. Utilizando este tipo de rejillas, podemos asegurar que el sitio web se redimensiona en función del espacio disponible, sin la necesidad de utilizar media query. Posteriormente es posible utilizar media query si es necesario realizar cambios

significativos en la estructura de la web.



Veamos cinco componentes que nos permitirán construir una rejilla flexible, y cómo utilizarlas:

- Fuentes flexibles
- Contenedores flexibles
- Imágenes flexibles

6.4.1 Fuentes flexibles

Cuando hablamos de diseño responsivo solemos dar más importancia al layout, pero el texto que vamos a presentar es igual de importante para conseguir un buen diseño.

Si no tenemos cuidado, nos podemos encontrar con varios problemas:

- Líneas cortas con pocos caracteres, lo que dificulta la lectura.
- Líneas largas con muchos caracteres, lo que también dificulta la lectura.
- Caracteres muy pequeños.

Lo ideal, según estudios, es una letra de un tamaño adecuado, para poder leer sin forzar la vista, y que se disponga formando **líneas de entre 60-80 caracteres**.

Para intentar solucionar esto con garantías lo más recomendable es utilizar para el texto unidades de tamaño relativas al viewport. Tendremos:

- **vw** en relación a la anchura del viewport.

- **vh** en relación a la altura del viewport.
- **vmin** el valor menor en relación a la dimensión pequeña del viewport (anchura o altura).
- **vmax** el valor máximo en relación a la dimensión más grande del viewport (anchura o altura).

Teniendo en cuenta, por ejemplificar que 1vw es un 1% de la anchura del viewport.

No obstante, encontrar siempre el tamaño perfecto de texto es algo que puede ser complicado sino imposible. Sin embargo allí van unas posibles **pautas**:

- Calcular el tamaño mínimo y máximo que me puedo permitir usando calc() (función css)
- No importa si el texto hace "reflow" (pasa a la siguiente línea) en determinadas ocasiones.
- Mantener el tamaño perfecto de línea puede que sea imposible.
- Si tenemos que elegir es conveniente priorizar móviles y tablets. ● Algunos elementos, por ejemplo en un menú de navegación, puede tener sentido usar tamaños fijos de letra en vez de unidades relativas al viewport.

6.4.2 Contenedores flexibles

Conociendo los valores objetivo, la primera tarea es establecer nuestro ancho base. Esta medida hace referencia al **ancho** que hemos definido a nuestro layout sin aplicar ningún diseño adaptativo. Como medida general, suele adoptarse como ancho base un 90% del área visible de la pantalla, lo que permite seguir manteniendo toda la estructura y contenidos de la web siempre visibles.

Tamaños máximos y mínimos

Si buscamos un cierto grado de control aún mayor, una opción recomendable sería recurrir a las propiedades **max-width y min-width**, con las que podemos indicar el ancho de un elemento como máximo y el ancho de un elemento como mínimo respectivamente, consiguiendo así garantizar cierto grado de control del diseño.

```
.contenedor_reponsive{
        background-color: #fff;
        width: 90%;
```



```
 max-width: 960px;
 width: min(90%, 960px); /* Equivalente a width: 90% y max-width: 960px */ height:
        400px;
        margin: 0 auto;
```

```
}
```

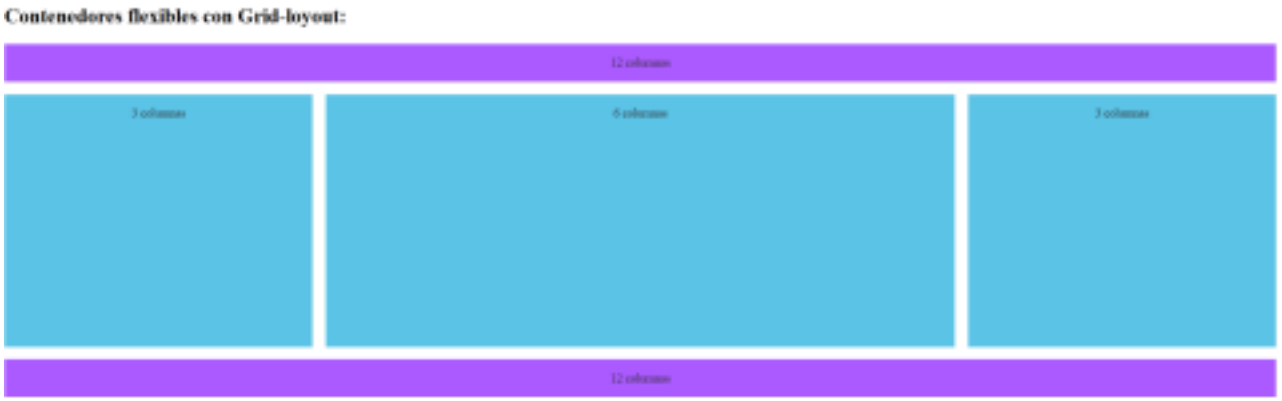
Ahora que tenemos nuestros contenedores flexibles, el siguiente paso es obtener el mismo comportamiento para los márgenes y rellenos.

Sin embargo, **utilizar porcentajes no nos garantiza un diseño adaptativo** de calidad, hay que comprender bien varios detalles. El primer problema que encontraremos será que si sumamos el tamaño de los elementos 70% + 30%, junto a los bordes de cada uno, 2px por cada lado, la suma es superior al 100% del contenedor padre, por lo que no cabe en su interior. El segundo elemento se desplaza a la zona inferior, descuadrando todo el diseño. Lo mismo puede ocurrir si intentamos añadir margin o padding.

Hay varias formas de **solucionarlo**:

- Eliminar los bordes y reducir los porcentajes manualmente hasta que quepan en el 100% del padre.
- Entender la propiedad **box-sizing** y cambiar el modo en el que se gestiona el modelo de cajas.
- **Opción ideal:** Utilizar un sistema moderno como **Flexbox o Grid**.

Ejercicio:



6.5 Imágenes flexibles

6.5.1 Imágenes fluidas

Aplicar la siguiente restricción a las imágenes:

```
img { max-width: 100%;
      height: auto; /* Mantiene la relación de aspecto original */
10
}
```

o utilizar:

```
background-image: url("img/sky.jpg");
 background-repeat: no-repeat;
 background-size: cover;
```

Las imágenes son un elemento fundamental de todas las páginas y representan una gran parte del peso de la misma. Esta situación plantea ciertos retos a la hora de hacer *diseño responsivo.

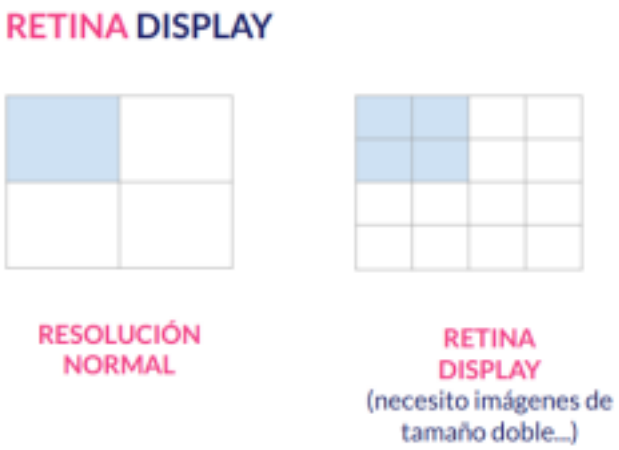
Estos retos son principalmente dos:

- Retos a nivel de optimización de la página web.
- Retos a la hora de diseñar la página.

6.5.2 Optimización de Imágenes en el Diseño Responsivo. Cuando nos referimos a optimizar el uso de las imágenes nos referimos a:

- Consumir el menor ancho de banda posible.
- Elegir la versión de una misma imagen más adecuada para la resolución. En este proceso de optimización debemos tener en cuenta:

- Ancho del dispositivo.
- Dimensiones de la imagen.
- Resolución de la imagen (en especial en los dispositivos RETINA DISPLAY) 11



Este tipo de dispositivos tienen una densidad de píxeles superior a la normal. En la imagen se muestra la explicación para dispositivos con densidad DPR 2x. En estos casos para que las imágenes se muestren bien tendrán que ser de tamaño doble.

6.5.3 Optimización. Imágenes SVG

Para solucionar este tipo de problemas lo más fácil es usar imágenes SVG que son gráficos vectoriales que escalan y encogen sin perder resolución.

El problema es que no siempre disponemos de gráficos SVG.

6.5.4 Optimización. Imágenes PNG/GIF/JPEG/WEBP/AVIF

En estos casos, si no nos importa la optimización es suficiente con usar una imagen de gran resolución y dimensiones y acotarla dentro de un contenedor. Por ejemplo:

```
<div><img src="".......""  /></div>

div {
 /* dimensiones deseadas */
}
img {
 max-width: XXXXXpx;
 width: 100%;
}
```

Pero sin embargo, si nos importa la optimización utilizaremos los atributos scrset y/o sizes de la imagen que queremos mostrar. Según “retina display” x1 x2 x4...

Por ejemplo:

```
<!-- Considerando resolución: retina display →

<div class="container">
 <img src="img/small.jpg" srcset="img/big.jpg 2x, img/small.jpg 1x" />
</div>

<!-- Considerando dimensiones: Unidades del contenedor →

<div class="container">
 <img
 src="img/small.jpg"
 srcset="img/big.jpg 2000w, img/small.jpg 1000w" />
</div>
```

6.5.5 Diseño “ART-DIRECTOR”

La técnica de diseño responsivo Art Director consiste en elegir una u otra imagen utilizando la etiqueta source dentro la etiqueta picture y sus atributos srcset para indicar la imagen y media que funciona de manera similar a una media query.

Se ve muy claro con un ejemplo:

```
<div class="art">

 <picture>

 <source media="(min-width: 576px)" srcset="img/big-art.jpg" />
<source media="(max-width: 575px)" srcset="img/small-art.jpg" />
<img src="img/small-art.jpg" />

 </picture>

</div>
```

Normalmente este diseño consiste en fotos que se acercan al objeto importante.

Al final, en casos reales siempre hay que combinar ambas técnicas y probar, en la medida de lo posible, en dispositivos reales, incluidos móviles con Retina Display.

6.6 Tablas flexibles

Las tablas son un elemento problemático a la hora de realizar Diseño Responsivo ya que en cuanto tienen un número de columnas considerable pueden provocar la aparición del tan temido scroll horizontal en nuestra página web, sobre todo en pantallas pequeñas.

Para afrontar este tipo de problemas hay tres soluciones que son aceptadas como buenas:

- Esconder columnas.
- Convertir las filas en listas
- Crear un scroll horizontal que solo se aplique a la tabla.

6.6.1 Esconder columnas

Esta técnica consiste básicamente en esconder ciertas columnas, que deben de ser las menos importantes, cuando el tamaño de la pantalla es menor que un breakpoint establecido.

Tiene las siguiente ventajas:

- Conseguimos un diseño responsivo.
- Priorizo el contenido que quiero mostrar.

Aunque también tiene desventajas:

- Pierdo información en determinadas pantallas.

Un ejemplo para conseguir esto sería una hoja de estilos similar a esta:

```
@media screen and (max-width: 576px) {
 .hidden td.primero,.hidden th.primero {
 display: none; }
}
@media screen and (max-width: 768px) {
 .hidden td.segundo, .hidden th.segundo {
```

```
 display: none; }
}
```

En ese diseño tenemos dos *breakpoints* y ocultaremos unas u otras columnas dependiendo

del tamaño (la que tienen la clase *.primero* la clase *.segundo*).

6.6.2 Convertir Listas a filas

Esta técnica consiste en hacer desaparecer las cabeceras de la tabla cuando la pantalla es menor que una determinada cantidad y hacer que todas las celdas se conviertan en elementos de bloque para que se muestran una debajo de otra y no al lado.

Tiene las siguiente ventajas:

- Conseguimos un diseño responsivo.
- No pierdo información.

Aunque también tiene desventajas:

- No priorizo la información.
- "Desplazar" mucho el resto del layout hacia abajo.

```
@media screen and (max-width: 700px) {
 table.listas,
 table.listas thead,
 table.listas tbody,
 table.listas tr,
 table.listas th,
 table.listas td {
 display: block;
 }
```

6.6.3 Scroll controlado

Esta técnica consiste en acotar el scroll horizontal para que si ha de aparecer solo afecte a la tabla y no a la página entera.

Tiene las siguiente ventajas:

- Conseguimos un diseño responsivo.
- No pierdo información.

15
Aunque también tiene desventajas:

- No priorizo la información.
- Aunque local, sigue habiendo un scroll horizontal.

Para conseguir esto debemos “envolver” la tabla en un contenedor y darle las siguiente propiedades:

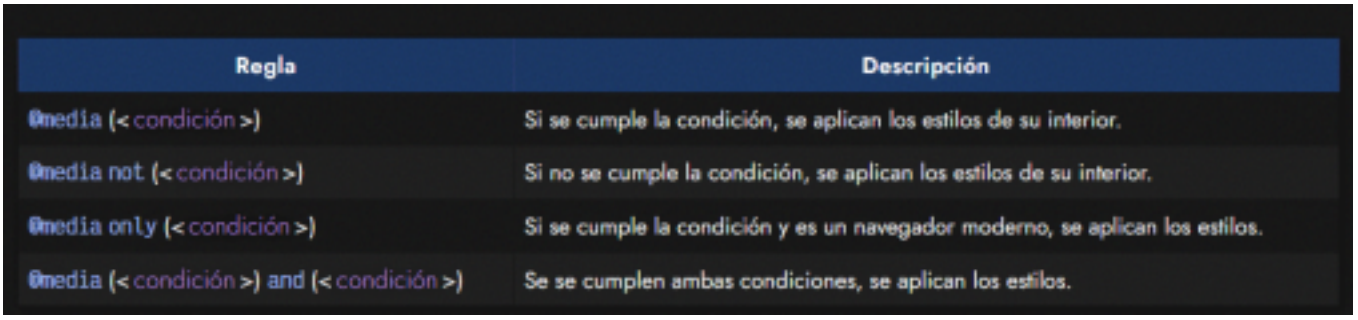
```
<div class="localscroll">
 <table> </table>
</div>
div.localscroll {
 overflow-x: auto;
 width: 100%;
}
```

No obstante no hay una solución universal. Debemos experimentar según el Layout que tengamos para ver cuál de estas técnicas (o la mezcla de ellas) se adecúa mejor a nuestro diseño.

6.7 Media Queries

Mediante **Media Queries**, podemos hacer excepciones para que unos determinados estilos de diseño sólo se apliquen si se cumplen una serie de condiciones, generalmente relacionadas con el dispositivo o navegador mediante el cuál se está viendo la página.

A fin de cuentas, se trata de una especie de condicionales de diseño, en las que si se cumple la condición se aplicarán unos estilos, y en caso contrario, se aplicarán otros.



Regla	Descripción
@media (<condición>)	Si se cumple la condición, se aplican los estilos de su interior.
@media not (<condición>)	Si no se cumple la condición, se aplican los estilos de su interior.
@media only (<condición>)	Si se cumple la condición y es un navegador moderno, se aplican los estilos.
@media (<condición>) and (<condición>)	Se se cumplen ambas condiciones, se aplican los estilos.

Así pues, veamos un pequeño ejemplo donde escribimos las dos opciones anteriores,

pero sin entrar en detalles aún en la condición:

```
@media (*condición*) {
        .container {
 background: green; }}

@media not (*condición*) {
        .container {
 background: red; }
```

En el ejemplo anterior, si se cumple la condición establecida, se aplicará un color verde. Sin embargo, si no se cumple, se aplicará un color rojo. Recuerda que es similar al funcionamiento de un if / else en programación.

No olvides que al escribir una regla @media podrías estar sobreescribiendo los estilos CSS en otro fragmento posterior. Una buena forma de empezar a escribir MQ sería escribir las reglas @media siempre al final, como excepciones al código anterior.

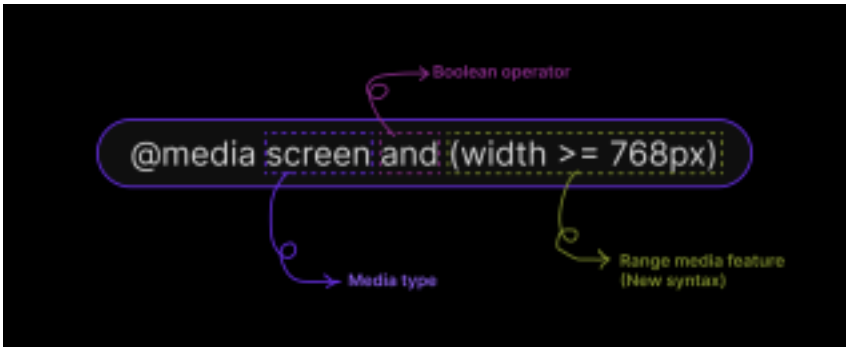
Si lo deseas, es posible establecer múltiples condiciones en las reglas @media. De esta forma, se pueden conseguir situaciones mucho más específicas y flexibles. Ten mucho cuidado si aplicas el not en las condiciones, no sea que niegues de forma incorrecta los casos deseados:

```
@media (*condición*) and (*condición) {
 .container {
 background: orangered;} }
```

6.8 Media Query Range Syntax

Aunque existen otras formas, hoy en día la forma preferida de escribir Media Queries es utilizando la modalidad de rangos de condiciones (Media Query Range Syntax), mucho más versátiles que las sintaxis anteriores, y mucho menos tediosas.

17



Para ello, vamos a escribir las condiciones utilizando operadores de comparación como **<, <=, > o >=:**

```
.menu {
 height: 100px;
}

@media (width <= 400px) {
 .menu {
 background: blue; }
}

@media (400px <= width <= 800px) {
 .menu {
 background: red; }
}

@media (width >= 800px) {
 .menu {
 background: green; }}
```

6.9 Patrones Responsive

Hasta ahora, hemos definido todas las herramientas necesarias para crear estructuras adaptativas: diseños basados en rejillas flexibles, estrategias para incorporar elementos multimedia y la potencia de los *Media Query* para adaptar el contenido a nuestras necesidades. Vemos cómo incorporar estas técnicas a la hora de desarrollar un sitio web adaptativo.

6.9.1 La importancia del contexto

Un diseño adaptativo, implementado de una manera correcta, puede dar a los

de la web, un alto nivel de continuidad entre los diferentes *contextos*. Esto es así, porque de la manera más simple, un diseño adaptativo es capaz de mostrar un documento HTML correctamente en una infinidad de dispositivos, gracias a una estructura flexible y los *Media Query* que aseguran un diseño portable y accesible en la manera de lo posible.

De alguna manera, es posible identificar el *contexto* en el que se visita una web, a partir del dispositivo utilizado. En este contexto, podemos definir un tipo de usuario y sus objetivos. En otras palabras, un usuario móvil quiere un acceso rápido a la información y realizar diferentes tareas a las que realizaría sentado en su sofá con su portátil. En este caso, el tiempo y el ancho de banda están en extremos totalmente opuestos.

Por otra parte, si las prioridades y los objetivos del usuario varían según el *contexto*, entonces puede que disponer de un único documento HTML pueda suponer un problema, dependiendo de la manera en la que se encuentre estructurada la información.

De todas formas, es complicado suponer el *contexto* del usuario únicamente por el tipo de dispositivo, ya que efectivamente, es solamente eso: **una suposición**. ¿Cómo diferenciar, si mi navegación *móvil* se realiza desde la entrada del metro o desde el sofá de mi casa? ¿Es posible por tanto suponer un *contexto*?

Por lo tanto, no podemos inferir el *contexto* del usuario en base a su dispositivo. Así mismo, las palabras *mobile* o *desktop* no definen el comportamiento en la que los usuarios acceden a la web: un portátil puede ser un dispositivo móvil (por ejemplo en

un tren), al igual que un *smartphone* o *tablet* puede estar fijo en un lugar. El desarrollo adaptativo no pretende ser un reemplazo de los actuales sitios móviles, sino que forma parte de una estrategia de desarrollo, donde se pretende evaluar si efectivamente es necesario separar totalmente la experiencia móvil o tiene más sentido mostrar la información de una manera adaptativa.

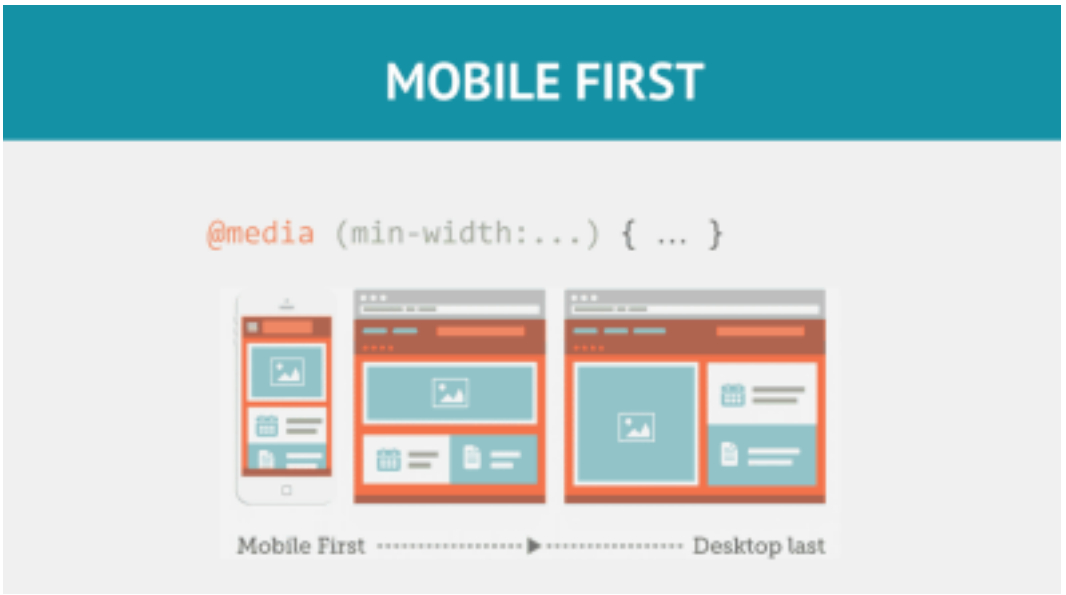
6.9.2 Hacia un flujo de trabajo adaptativo

Una de las primeras preguntas (si no la primera) que debemos hacernos, a la hora de plantearnos un diseño adaptativo es la siguiente: **¿En qué medida este contenido o funcionalidad beneficia o aporta valor a nuestros usuarios?.**

Esto es algo que deberíamos preguntar **siempre** para cualquier tipo de proyecto, sea web o no.

Si partimos de un planteamiento *mobile first*, hay que asegurarse que la información que mostramos, y las funcionalidades que implementamos sean importantes para el usuario, ya que no hay espacio suficiente para todo. Hay que darse cuenta de lo que realmente importa. Diseñar partiendo de este paradigma, nos obliga a concentrarnos en lo realmente importante.

19



*If you design mobile first, you create agreement on what matters most. You can then apply the same rationale to the desktop/laptop version of the web site. We agreed that this was the most important set of features and content for our customers and business; why should that change with more screen space?*Luke Wroblewski

En otras palabras, diseñar desde un inicio pensando en dispositivos móviles puede enriquecer la experiencia de los usuarios, proporcionando un elemento que normalmente se pasa por alto: enfocarnos en lo realmente importante. Esto no quiere decir que los diseños sean simples, faltos de contenidos o funcionalidades, sino que

debemos concentrarnos en lo realmente importante.

6.9.3 Puntos de ruptura

El siguiente paso es identificar el número de diseños diferentes que vamos a crear, para acomodarnos a los distintos tamaños de dispositivos.

En las **estadísticas globales** anónimas de Global StatCounter, nos podemos hacer una idea de los atributos más comunes.

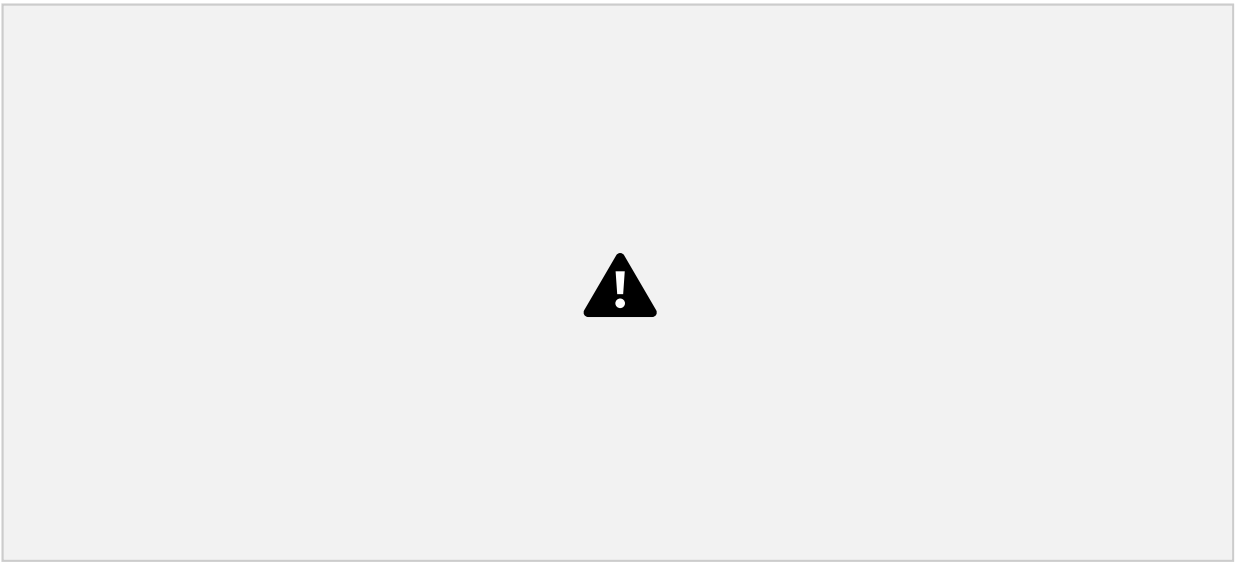
La siguiente tabla muestra los anchos más comunes a la hora de identificar los puntos de ruptura:

Punto de ruptura Dispositivo objetivo

- 320 pixels para dispositivos pequeños como teléfonos, en disposición vertical. ● 480 pixels para dispositivos pequeños como teléfonos, en disposición horizontal.

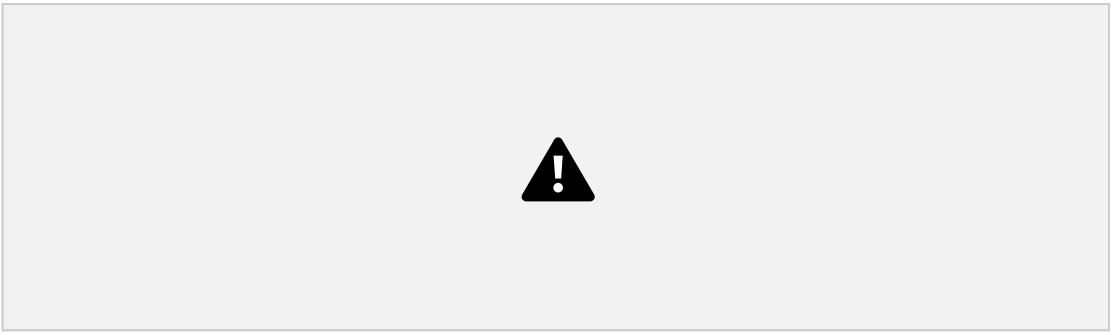
- 600 pixels para tabletas de menor tamaño, como Amazon Kindle (600×800), en disposición vertical.
- 768 pixels para tabletas de unas 10", como el iPad (768x1024), en disposición vertical.
- 1024 pixels para tabletas de unas 10", como el iPad (768x1024), y pequeños portátiles o*netbooks*, en disposición horizontal.
- 1200 pixels para pantallas panorámicas, en portátiles o dispositivos de escritorio.
- 1600 pixels para pantallas panorámicas, en portátiles o dispositivos de escritorio.

Puntos de ruptura en Bootstrap:



6.9.4 Patrones responsive

Afrontar un proyecto que requiera el desarrollo de una página web responsiva no es fácil y hay varias estrategias posibles para afrontarlo. No obstante ya es comúnmente aceptado que lo más recomendable es lo siguiente:



Es decir, que el proceso de diseño de nuestra página web debe empezar haciendo que todo

quede correctamente en pantallas pequeñas. De este manera:

- Priorizo siempre lo que es importante. Si el proceso fuera al revés es muy fácil que caigamos en el error de quitar cosas que sean realmente importantes. ● Me pregunto en cada diseño si es necesario un diseño nuevo para pantallas más grandes.
- Elijo los breakpoints más adecuados.

6.9.5 Patrones Responsivos

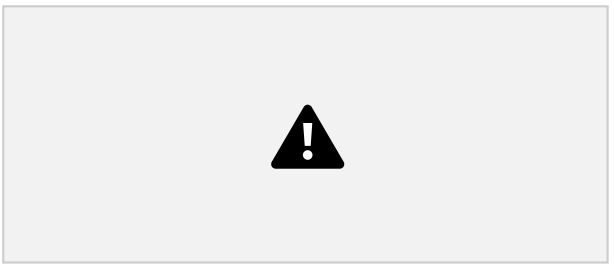
En nuestro proceso de diseñar páginas web responsivas, debemos conocer lo que se conoce como *patrones responsivos*.

Los *patrones responsivos* son soluciones que se han dado por buenas para el problema de diseñar páginas web responsivas. Hay muchos pero los más comunes son lo siguientes:

- Column Drop
- Mostly Fluid
- Layout Shifter
- Off Canvas
- Mezclar varios de ellos
- Pequeños ajustes (Tiny Tweaks)

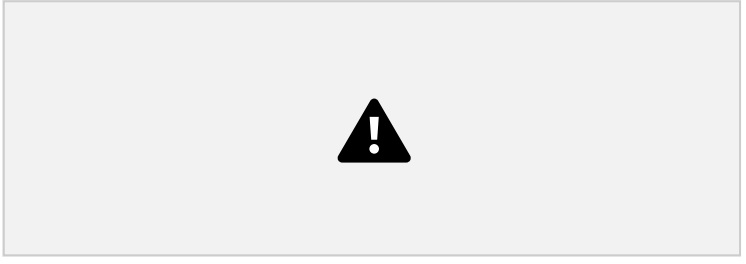
A continuación pasamos a comentar los principales, los cuatro primeros para los que hay un ejemplo en esta misma carpeta de este repositorio:

Column Drop

Es el patrón más básico y consiste en que en cada breakpoint se va a apilando un elemento: 

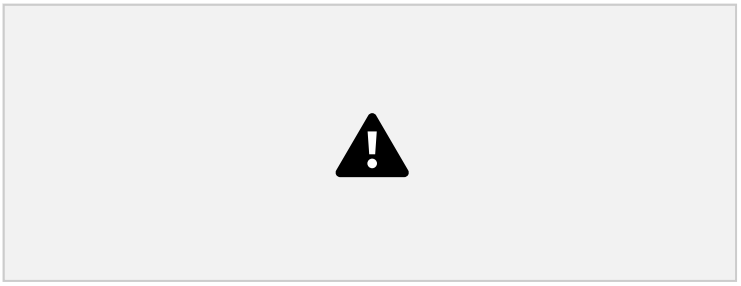
Mostly Fluid

Parecido a Column Drop. Es una cuadrícula fluida y en cada breakpoint hay redimensionamiento de varias columnas.



Layout Shifter

Es el patrón más responsivo, en cada breakpoints cambia el diseño del layout, no únicamente el flujo y la anchura de los elementos.

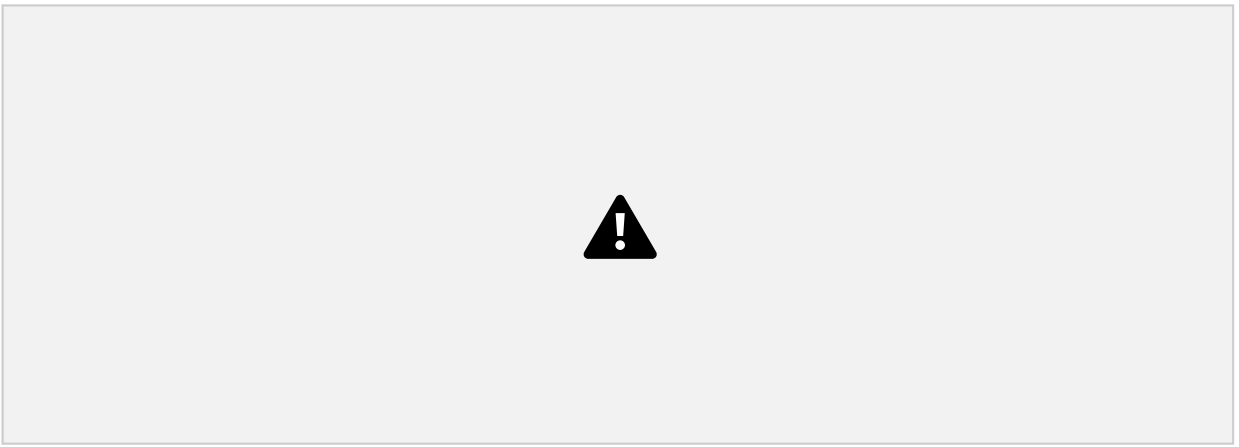


Off Canvas

En vez de apilar contenidos éstos se colocan fuera de la pantalla cuando el tamaño de pantalla no es lo suficientemente grande.

Practicar es la clave, [FrontMentor]

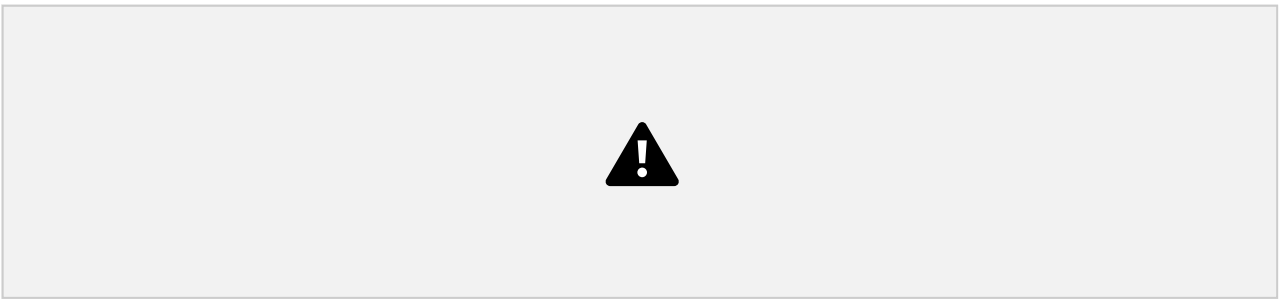
23



6.10 Tipos de medios

En algunas ocasiones, queremos indicar que las reglas @media sólo se pongan en funcionamiento en determinados tipos de dispositivos. Son los llamados tipos de medios, que

pueden utilizarse en las condiciones de los media queries. Existen los siguientes:



```
/* Pantallas menores de 600px */

@media screen and (width <= 600px) {
 /* ... */}

/* Previsualizaciones y formatos de impresión */

@media print {
 /* ... */}
```

6.11 Media Queries desde HTML

Por último, hay que tener en cuenta que los media queries también es posible indicarlos desde HTML, utilizando la etiqueta <link> y el atributo media para establecer la condición:

```
<link rel="stylesheet" href="mobile.css" media="(max-width: 640px)">

<link rel="stylesheet" href="tablet.css" media="(min-width: 640px) and
```

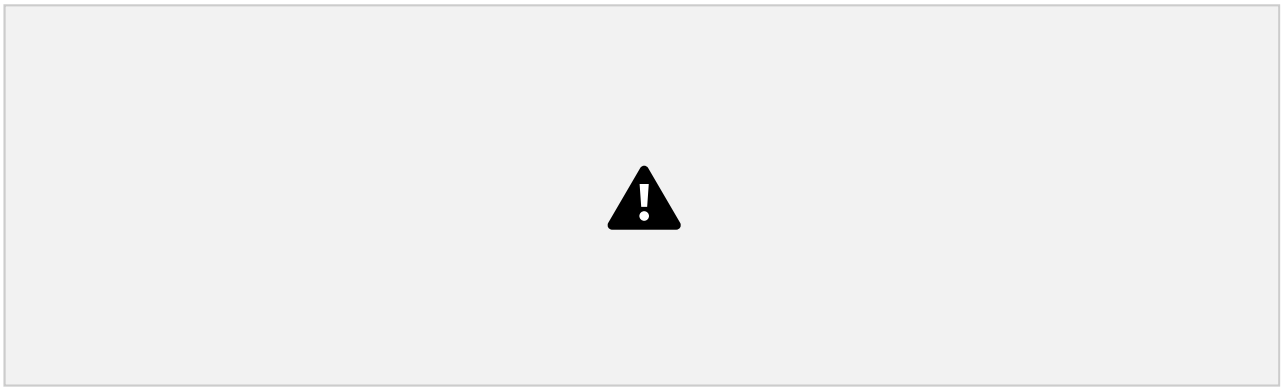


```
 (max-width: 1280px)">

<link rel="stylesheet" href="desktop.css" media="(min-width: 1280px)">
```

6.12 Preferencias de usuario

Hay que saber que las **media queries** tienen un apartado especial donde podemos utilizar preferencias de usuario. Las preferencias de usuario son reglas @media especiales que nos permiten dar estilo según la preferencia establecida por el usuario en su sistema operativo, adaptándose a su caso particular.



Dark mode / Light mode

Una de las características más recurrentes en interfaces de usuario es la posibilidad de elegir un dark mode o dark theme, es decir, un sistema que permita al usuario seleccionar un tema claro (generalmente con fondo blanco) o un tema oscuro (generalmente con fondo negro).

Aunque podemos hacer esto de forma manual, existe una regla @media especial denominada prefers-color-scheme donde podemos detectar si el usuario tiene preferencia por uno de estos dos valores (establecido en las opciones del sistema operativo) y actuar en consecuencia.

Ejemplo:

```
body {
 background-color: white;
 color: black;
 font-family: sans-serif;
 padding: 2rem;
 }

 /* Estilo para modo oscuro */

@media (prefers-color-scheme: dark) {

 body {
 background-color: #121212;
 color: #e0e0e0;}
```


Nota: En **Windows**, ve a Configuración > Personalización > Colores y elige Oscuro en "Elige

el modo de aplicación predeterminado".