Talento Digital para Chile:

Módulo 1 Fundamentos del Desarrollo Web

1.2.- Contenido 2: Estructurar contenido en una página web responsiva básica utilizando HTML y CSS para que se adapte a distintos dispositivos acorde a las buenas prácticas de la industria

Objetivo de la jornada

1. Codifica una página web que se adapta a distintos tipos de dispositivos utilizando los principales elementos de HTML y CSS para resolver una necesidad dada.
2. Verifica el funcionamiento de la página web utilizando las herramientas de desarrollador del navegador para verificar el despliegue de la página en distintos tipos de dispositivos.

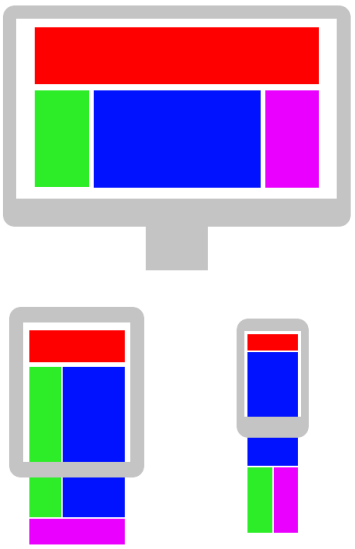
1.2.1.- Responsividad

1.2.1.1. Qué es Responsividad y para qué sirve

La diversidad de dispositivos que actualmente se utilizan para navegar en la web es progresivamente más creciente. Computadores de escritorio, laptops, tablets, televisores, smartphones y hasta relojes actualmente se cuentan entre las herramientas a través de las cuales los usuarios acceden a los contenidos de la web.

Estos dispositivos, además de sus diversas capacidades de procesamiento y almacenamiento, a menudo tienen muy distintos tamaños de la pantalla y requieren un enfoque diferente de cómo se presenta el contenido en la ésta.

El diseño web responsivo (O adaptativo), fue definido originalmente por Ethan Marcotte[1], es una respuesta a las necesidades de los usuarios y los dispositivos que ellos están utilizando. El diseño cambia según el tamaño y las capacidades del equipo. Por ejemplo, en un teléfono los usuarios verían el contenido que se muestra en una vista de una sola columna; y por otro lado, un tablet puede mostrar el mismo contenido en dos columnas. Además de esto, los tamaños de pantalla cambian constantemente, por lo que es importante que los sitios web puedan adaptarse a cualquier tamaño de pantalla, hoy o en el futuro. Los dispositivos tienen diferentes prestaciones, en base a las cuales el usuario interactúa con ellos, como por ejemplo, visitantes que utilizan un mouse para interactuar con el contenido y otros las funcionalidades táctiles de su hardware.



Concepto de adaptación de contenido de un sitio web en computador de escritorio, tablet y smartphone.

Para muchos sitios ofrecer una versión móvil ya no es una opción secundaria ni sirve decir "por ahora no es necesario", ya que los dispositivos móviles como tablets y smartphones cada vez son más utilizados para navegar.

Anteriormente, el problema de la diversidad de dispositivos se resolvía detectando qué navegador estaba empleando el dispositivo del cliente, ya sea mediante JavaScript o mediante un lenguaje de lado servidor, para posteriormente proveer la versión del sitio web apropiada o cargar una hoja de estilo específica. Esto obligaba a desarrollar varias versiones del sitio web. Años atrás, muchos sitios poseían su versión móvil, m.dominio.com y su versión tablet tablet.dominio.com, además de su tradicional www.dominio.com, y alojaban las distintas versiones en estos subdominios.

A diferencia de lo anterior, con el diseño web responsivo se cubren todas las resoluciones de pantalla con una sola versión en HTML, CSS y JS.

Podemos destacar las siguientes ventajas relacionadas con diseñar un sitio bajo la filosofía de diseño responsivo, tales como:

- Mejora en la experiencia de usuario, ya que permite una navegación agnóstica al dispositivo utilizado.

- Optimización de esfuerzos, costos de desarrollo y consistencia técnica, ya que pasamos de tener un solo sistema desarrollado y puesto en producción.
- Mejor SEO (Optimización en Motores de Búsqueda), por la existencia de una URL unificada, evitando redirecciones y los errores asociados a éstas.

Desde el punto de vista más técnico, específicamente en términos de códigos HTML y CSS como los que introdujimos en contenidos anteriores, debemos incorporar ciertas sentencias que nos permitirán manejar los conceptos de responsividad.

Etiqueta <meta>

Las páginas optimizadas para una variedad de dispositivos deben incluir una etiqueta de meta viewport en el encabezado del documento. Una etiqueta **meta viewport** le da al navegador instrucciones sobre cómo controlar las dimensiones y la escala de la página. Viewport es el tamaño de la ventana del navegador web.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    …
    <meta name="viewport" content="width=device-width, initial-scale=1">
    …
  </head>
  …
```

El uso del valor de **meta viewport width = device-width** le indica a la página que ajuste el ancho de la pantalla in pixeles independientes del dispositivo. Un píxel independiente del dispositivo es una representación de un solo píxel, que en una pantalla de alta densidad puede ser equivalente a un conjunto de píxeles físicos. Esto permite que la página redistribuya el contenido para que coincida con diferentes tamaños de pantalla, ya sea en un teléfono móvil pequeño o en un monitor de computador de escritorio[2].

Básicamente, lo que hace esta etiqueta es ajustar el ancho del documento al ancho del dispositivo e indicar que inicialmente no debe hacer zoom.

Los parámetros posibles de definir son los siguientes[3]:

- **width:** ancho de la página.
- **height:** alto de la página.
- **initial-scale:** escala o zoom inicial de la página.
- **minimum-scale:** escala o zoom mínimo que podemos hacer en la página.
- **maximum-scale:** escala o zoom máximo que podemos hacer en la página.

- **user-scalable:** establece si está permitido o no hacer zoom (yes | no).

Notas:
Las medidas se pueden establecer en pixeles o como device-width para que utilice el ancho disponible. El valor de los parámetros de escala indica el zoom, por ejemplo, un valor de 2.5 produce un zoom del 2,5 de aumento. Los parámetros van separados por comas.

Imágenes Responsivas
Es posible instruir, a través de nuestro código HTML y CSS, que las imágenes incluidas en nuestra página se adecúen a los tamaños disponibles según las dimensiones del dispositivo que esté navegándola.

Podemos utilizar la propiedad **width** con la que podemos especificar el ancho de nuestra imagen. Si definimos que width es de 100%, la imagen se ajustará al espacio que tenga el elemento que la contenga, dentro del código HTML, por ejemplo un elemento ** **[4].

```
<img src="img_girl.jpg" style="width:100%;">
```

Si deseamos evitar que la imagen sea escalada a dimensiones mayores a su tamaño original, para evitar visualizaciones pixeladas de la misma, podemos utilizar la propiedad **max-width**, con la que podremos limitar el máximo tamaño de ésta a su tamaño original.

```
<img src="img_girl.jpg" style="max-width:100%;height:auto;">
```

Finalmente, una técnica más sofisticada corresponde a definir que se cargue un archivo de imagen distinto dependiendo del ancho disponible para ésta según el dispositivo y elemento donde está emplazada. Para esto utilizamos la etiqueta **<picture>** de la siguiente manera, definiendo puntos de quiebre para cambio de una imagen a otra según el número de pizeles de ancho:

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
  <source srcset="flowers.jpg">
  <img src="img_smallflower.jpg" alt="Flowers">
</picture>
```

Tamaño responsivo para el texto

De la misma forma que para las imágenes, podemos indicar en nuestros código, que los textos posean un tamaño proporcional al tamaño de nuestro viewport, o tamaño de la ventana del navegador. En este caso se define una unidad que es relativa al viewport y que se denomina **vw** y corresponde a un 1% del ancho del viewport. Si el viewport posee un ancho de 50cm, entonces la unidad vw corresponderá a 0.5cm. La sintaxis para utilizar este tipo de unidad para el tamaño del texto es:

```
<h1 style="font-size:10vw">Hello World</h1>
```

Existen otras unidades relativas que pueden utilizarse para definir tamaño de los textos dentro de un archivo HTML utilizando CSS, tales como[5]:

Unidad	Descripción
em	Relativa al tamaño de la fuente del elemento (2em significa 2 veces el tamaño de la fuente actual)
ex	Relativa a la altura de la fuente actual
ch	Relativa al ancho del “0” (Cero)
rem	Relativa al tamaño de la fuente del elemento raíz
vw	Relativa al 1% del ancho del viewport
vh	Relativa al 1% de la altura del viewport
vmin	Relativa al 1% de la menor dimensión del viewport
vmax	Relativa al 1% da la mayor dimensión del viewport
%	Relativa al elemento de jerarquía superior

1.2.1.2. Tipos de dispositivos y orientaciones

En la actualidad existe una infinidad de dispositivos en las distintas categorías que hemos venido mencionando en las secciones previas. Es imposible contar con un listado completo de dispositivos y sus tamaños, para luego desarrollar cubriendo todos los posibles escenarios que enfrentará nuestra aplicación web cuando sea puesta en producción. Una lista completa necesitaría una actualización incluso diaria, por lo que está fuera de nuestras expectativas.

Si deseamos tener una idea de cuál es la distribución de tamaños de pantallas en pixeles en la actualidad, de los dispositivos que se usan para acceder a los contenidos de la web, debemos remitirnos a la siguiente distribución:

Escritorio y Laptops

Tamaño [Pixeles]	Porcentaje
1366×768	23.49%
1920×1080	19.91%
1536×864	8.65%
1440×900	7.38%
1280×720	4.89%
1600×900	4.01%
1280×800	3.33%
1536×864	8.65%

Tablets

Tamaño [Pixeles]	Porcentaje
768×1024	51.98%
1280×800	7.11%
800×1280	5.34%
601×962	4.47%
600×1024	2.85%
1024×1366	1.96%

Smartphones

Tamaño [Pixeles]	Porcentaje
360×640	17.91%
375×667	7.61%
414×896	6.52%
360×780	5.56%
360×760	5.06%
414×736	3.74%

Fuente: https://www.hobo-web.co.uk/best-screen-size/#chartHeaderTitle

Una herramienta de utilidad para investigar resoluciones de los distintos dispositivos del mercado es: https://www.screensizemap.com/, donde explorando con el puntero del mouse podremos navegar por marcas, modelos y resoluciones.

Debemos considerar que en el caso de Tablets y Smartphones, por motivos de orientación, cada resolución debe considerarse en ambos sentidos.

1.2.1.3. El concepto Mobile First

Mobile First es una filosofía que nos entrega una forma de enfrentar el trabajo de diseños de web responsivas, que tiene como precepto comenzar teniendo en mente los dispositivos con pantallas de menor tamaño. Así, **Mobile First** es en realidad un concepto bastante simple: diseñar pensando en los móviles primero.

Contenido

Por razonas de espacio, no es posible tener la expectativa de mostrar tanta cantidad de contenido en un dispositivo móvil como en uno de escritorio. Por esto, antes incluso de considerar los aspectos estéticos, debemos contextualizar el volumen de información a algo que sea efectivamente factible de mostrar en un móvil.

Bajo la filosofía de Mobile First consideraremos com tarea inicial la confección de la especificación del contenido de la página. Esta especificación debe contener simplemente un listado (En palabras) de todos los contenidos que queremos tratar en una página, ya sea la principal o una de alguna sección interna. Para esto debemos tener en cuenta permanentemente el esquema móvil y su pantalla de dimensiones reducidas, que no es capaz de desplegar una cantidad voluminosa de información.

La especificación del contenido no se refiere a nada gráfico, ni a esquemas de información, jerarquías o secciones del sitio en el árbol de contenido. Es simplemente un listado de elementos que podríamos escribir con palabras en un cuaderno. Es un

inventario de elementos, algo como logotipo, buscador, navegador con las principales secciones, imagen con el producto principal o los 3 productos principales, etc.

La definición de elementos debemos llevarla a cabo con el cliente y seguramente inicialmente se considerarán muchos contenidos que finalmente se concluirán como prescindibles. Al pensar en elementos como logotipo, barra de navegación, login y buscador, debemos asumir que estarán en una versión iconizada que eventualmente se muestren al tocarlas aprovecharndo las funcionalidades táctiles.

Diseño

La etapa creativa, en bosquejos de prototipo o en la implementación HTML + CSS, también se lleva a cabo teniendo primeramente en mente el esquema que sería visible en un escenario de dispositivo móvil. Resulta mucho más sencillo diseñar una web para un móvil, que tiene muy pocos elementos, que para un equipo de escritorio que posee mucho espacio para mostrar una infinidad de contenido. Así, es una buena práctica, considerar como inicio un espacio reducido y evaluar qué contenidos son razonables para este espacio y cuales pueden ser prescindibles.

En caso de ya estar abordando la implementación del código HTML + CSS, será útil reducir la ventana del navegador de previsualización a las dimensiones del ancho de un smartphone, por ejemplo 360 pixeles (Según lo expuesto en la seccion 1.2.1.2). En la primera aproximación es posible que se evidencie la necesidad de prescindir de más elementos que los pensados inicialmente.

Como desarrollador, es útil tener la posibilidad de revisar desde nuestro smartphone el estado del desarrollo y probar la experiencia de navegación. Existen herramientas web y también facilidades en las herramientas de desarrollador de nuestro navegador, para revisar el aspecto de nuestras páginas en distintos tamaños de dispositivo.

El primer objetivo es llegar a diseño satisfactorio para el dispositivo de menor tamaño, y con esto comenzar visualizar (Con alguna herramienta de desarrollo) el aspecto de nuestra página en anchos de pantalla progresivamente más amplios. La idea de este proceso es encontrar y definir, en base a criterios visuales, los puntos de quiebre (Brakpoints) en términos de resolución en los que debamos tomar una decisión de ajuste de la disposición de nuestros contenidos. Para lograr esta adaptación utilizaremos **Media Queries** (Siguiente sección), que corresponden prestaciones de CSS para implementar los mencionados breakpoints de resolución en nuestro código[6].

La filosofía Mobile First está de acuerdo a una concepción de nuestro sitio o página web desde menos a más. Éste será un proceso iterativo donde progresivamente se irán ajustando contenidos y disposición de elementos.

1.2.1.4. Utilización de Media Query

Adicionalmente a las técnicas de adaptación del navegador para mostrar una página web considerando las técnicas descritas en la sección 1.2.1.1., en las que vimos la forma de instruir al navegador para mostrar el contenido en una cierta escala inicial con la instrucción **meta viewport**, o ajustar contenidos específicos como imágenes y tamaño de textos, existe otro método más sofisticado, a través de las llamadas **Media Queries**, que nos permiten definir estilos totalmente distintos dependiendo de las dimensiones de nuestro navegador.

Lo anterior se logra con Media Queries, a través del uso de la regla **@media** que en concreto permite incluir propiedades CSS en la hoja de estilos sólo en el caso de que cierta condición se verifique. Por ejemplo, que la ventana de nuestro navegador sea de un ancho menor o mayor a un tamaño determinado. Así, escribiríamos el siguiente código para instruir con un breakpoint según lo explicado en la sección anterior. En este caso ordenamos que, si el dispositivo despliega la página web en un ancho de pantalla de navegación menor o igual a 600px, el color de fondo del elemento **<body>** sea rojo[7]**:**

```
@media only screen and (max-width: 600px) {
  body {
    background-color: red;
  }
}
```

Este es sólo un ejemplo ilustrativo para comprender el concepto detrás de @media. En la práctica se pueden incluir muchas sentencias de este tipo, para hacer reaccionar el navegador frente a distintas configuraciones, ancho del viewport y orientación del dispositivo. En las siguientes secciones veremos Media Queries desde un punto de vista más aplicado para lograr responsividad de varias características.

1.2.1.5 Responsividad con o sin la ayuda de un framework

Como se puede intuir, además de adaptaciones responsivas de elementos individuales como imágenes y tamaños de fuentes como según vimos en 1.2.1.1, existe una

infinidad de otras consideraciones y elementos que deben adaptarse al momento de pensar en una página responsiva, tales como menús, barras de navegación, encabezados, pié de página, etc. Por ejemplo, es usual encontrarse con páginas web que poseen diversas columnas o secciones al observarlas en un computador de escritorio, pero que al accederlas desde un móvil se despliegan ordenadas verticalmente una después de otra; asimismo las barras de navegación horizontales en modo escritorio, en dispositivos móviles se despliegan como un botón que se expande al tacto.

Todo lo anterior puede abordarse de dos formas:

1. Programando en CSS puro, incluyendo Media Query, todos los casos de adaptaciones de estilo para cada uno de los elementos de nuestro proyecto.
2. Utilizando un Framework o librería existente, confeccionada por terceros, y adoptándola como parte de nuestro desarrollo. Entre estos frameworks se cuentan algunos como la hoja de estilos W3.CSS o el set Bootstrap. El primero utiliza sólo CSS para lograr responsividad; y el segundo es más elaborado, basándose en CSS y la librería JQuery de Javascript para obtener resultados responsivos y efectos dinámicos de adaptación.

Cubriremos el framework Bootstrap en un capítulo posterior de este curso, por lo que en este caso revisaremos el caso correspondiente al punto 1 recién mencionado. Según esto, mostraremos a continuación una forma de implementar un sistema llamado Grid View, que equivale a cuadrícular nuestra área de trabajo en una página web, cuyos cuadros se redistribuyan de acuerdo al ancho de nuestra ventana de navegador.

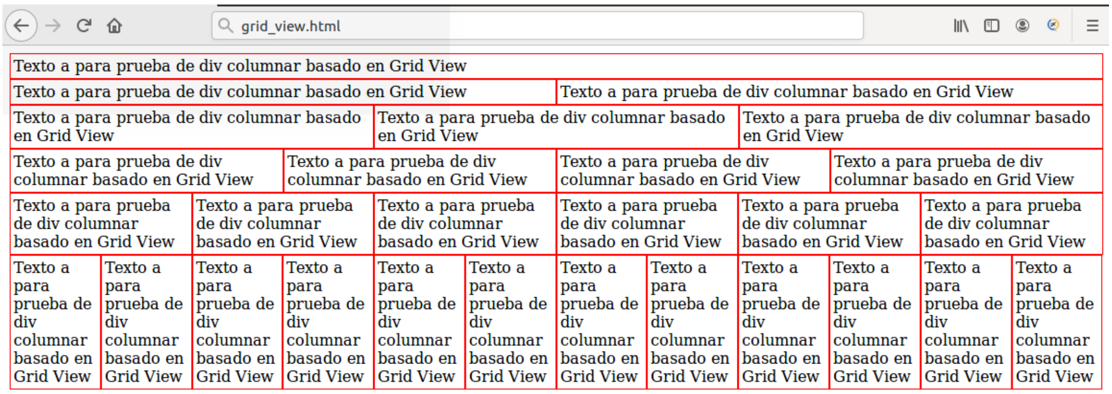
Primero, dividiremos el 100% del ancho del viewport en 12 columnas, que es usual para diseños responsive. De esta forma cada columna tendrá un 8.33% del ancho total. Y de esa forma generaremos clases para elementos HTML, generalmente elementos ** **, que les permitirán tener anchos de tamaños dados por múltiplos enteros de la columna unitaria de 8.33%. Así agruparemos, conjuntos de estos elementos dentro de un ** ** del tipo fila, que configuraremos como ** **. De esta forma podremos diagramar fácilmente columnas y filas para nuestras páginas.

El ejemplo corresponde al siguiente código:

Archivo: grid_view.html

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>
            * {
                box-sizing: border-box;
            }
            .col-1 {width: 8.33%;}
            .col-2 {width: 16.66%;}
            .col-3 {width: 25%;}
            .col-4 {width: 33.33%;}
            .col-5 {width: 41.66%;}
            .col-6 {width: 50%;}
            .col-7 {width: 58.33%;}
            .col-8 {width: 66.66%;}
            .col-9 {width: 75%;}
            .col-10 {width: 83.33%;}
            .col-11 {width: 91.66%;}
            .col-12 {width: 100%;}
            [class*="col-"] {
                float: left;
                padding: 3px;
                border: 1px solid red;
            }
            .row::after {
                content: "";
                clear: both;
                display: table;
            }
        </style>
    </head>
    <body>
        <div class="row">
            <div class="col-12">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-6">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-6">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-4">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-4">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-4">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-3">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-3">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-3">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-3">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
    </body>
</html>
```

El resultado de esta construcción será una visualización como la que se muestra a continuación:



Sin embargo, a pesar de tener una vista de columna en base a cuadrículas, aún nuestra página no se adapta frente a cambios del ancho de nuestra ventana de navegación y se muestra de la misma forma independiéntemente del tamaño del dispositivo. Para lograr esto y mostrar los conceptos tratados hasta el momento, hacemos uso de **Media Query** y el concepto de **Mobile First**, según lo explouesto en las secciones anteriores. La regla **@media** nos permitirá definir un breakpoint que, para fines didácticos definiremos de manera muy simple.

Podríamos instruir al navegador a que, ajuste todos los anchos los elementos **<div>**, de cualquier clase **.col-**, a un 100% si el ancho de la ventana del navegador es igual o menor a 768px. Esto lo haríamos utilizando **@media only screen and (max-width: 768px)**. A pesar de consistir en una lógica correcta, no estaríamos partiendo desde el concepto Mobile First, por lo que pensaremos la situación en una forma inversa. Estableceremos que, por defecto, todos los elementos <div> de cualquier clase **.col-** poseen un ancho del 100% de la ventana de navegación. De esta forma, estamos pensando en nuestro layout móvil en primera instancia. Luego de esto, utilizando **@media only screen and (min-width: 768px)**, daremos instrucciones de cómo diagramar la página en formato de columnas, usando la Grid View anterior, en caso que la ventana de navegación tenga un ancho igual o mayor a 768px. Esto será una adaptación desde el caso por defecto de vista móvil.

De esta forma el código utilizando **Media Query** y el concepto de **Mobile First** será:
Archivo: responsive.html

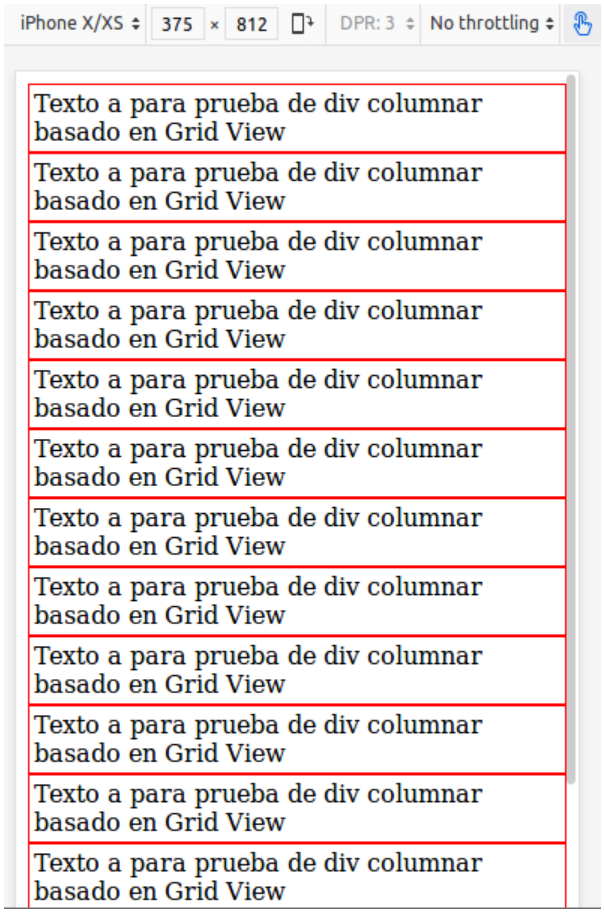
```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>
```

```
            * {
                box-sizing: border-box;
            }

            /* Para versión móvil (Mobile First): */
            [class*="col-"] {
                width: 100%;
                padding: 3px;
                border: 1px solid red;
            }

            /* Para versión escritorio o laptop: */
            @media only screen and (min-width: 768px) {
                .col-1 {width: 8.33%;}
                .col-2 {width: 16.66%;}
                .col-3 {width: 25%;}
                .col-4 {width: 33.33%;}
                .col-5 {width: 41.66%;}
                .col-6 {width: 50%;}
                .col-7 {width: 58.33%;}
                .col-8 {width: 66.66%;}
                .col-9 {width: 75%;}
                .col-10 {width: 83.33%;}
                .col-11 {width: 91.66%;}
                .col-12 {width: 100%;}
                [class*="col-"] {
                    float: left;
                }
                .row::after {
                    content: "";
                    clear: both;
                    display: table;
                }
            }
        </style>
    </head>
    <body>
        <div class="row">
            <div class="col-12">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-6">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-6">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-4">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-4">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-4">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-3">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-3">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-3">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-3">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-2">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
            <div class="col-1">Texto a para prueba de div columnar basado en Grid View</div>
        </div>
        <div class="row">
            <p> Texto posterior a la grilla de columnas</p>
        </div>
    </body>
</html>
```

En este caso si vemos nuestro resultado en una pantalla, por ejemplo de un iPhone X con un viewport de 375 x 812px, tendremos un resultado como el siguiente:



En este caso, puesto que el viewport es 375px, menor al breakpoint 768px, todos los elementos se muestran con un ancho del 100% de la ventana de navegación.
Con las tablas y la herramienta web https://www.screensizemap.com/, entregadas en la sección 1.2.1.2, puede experimentarse con diferentes puntos de quiebre según viewport del dispositivo, además confeccionando distintos estilos para distintos rangos de estos.

Sugerimos al lector investigar a través de experimentación cuál es el efecto de no incluir la propiedad **box-sizing**, y su efecto en el ejemplo responsivo recién expuesto.

Finalmente, otro uso interesante de Meria Query es la detección de la orientación de la pantalla y a aplicación de distintas propiedades CSS según esto. Una regla @media para este tipo de casos sería:

```
@media only screen and (orientation: landscape) {
  body {
    background-color: red;
  }
}
```

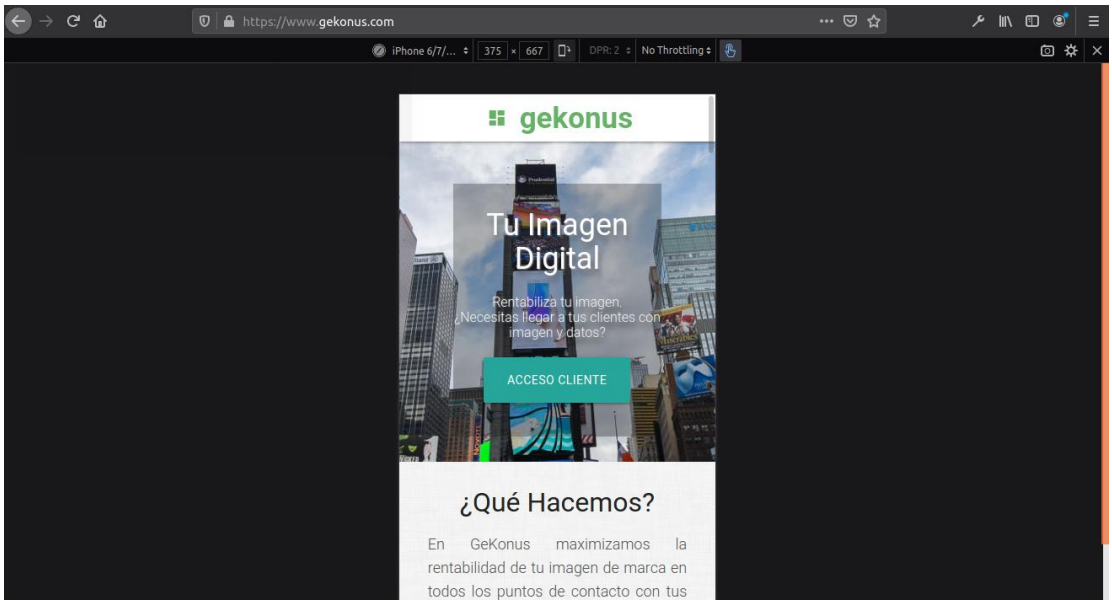
Este ejemplo pinta de color rojo el fondo del elemento <body> de nuestro documento HTML en caso que la ventana de navegación corresponda a un dispositivo que está en posición horizontal.

Existen otras variedades de configuraciones posibles de implementar utilizando CSS y Media Queries, que pueden ser exploradas en las referencias de este documento[7].

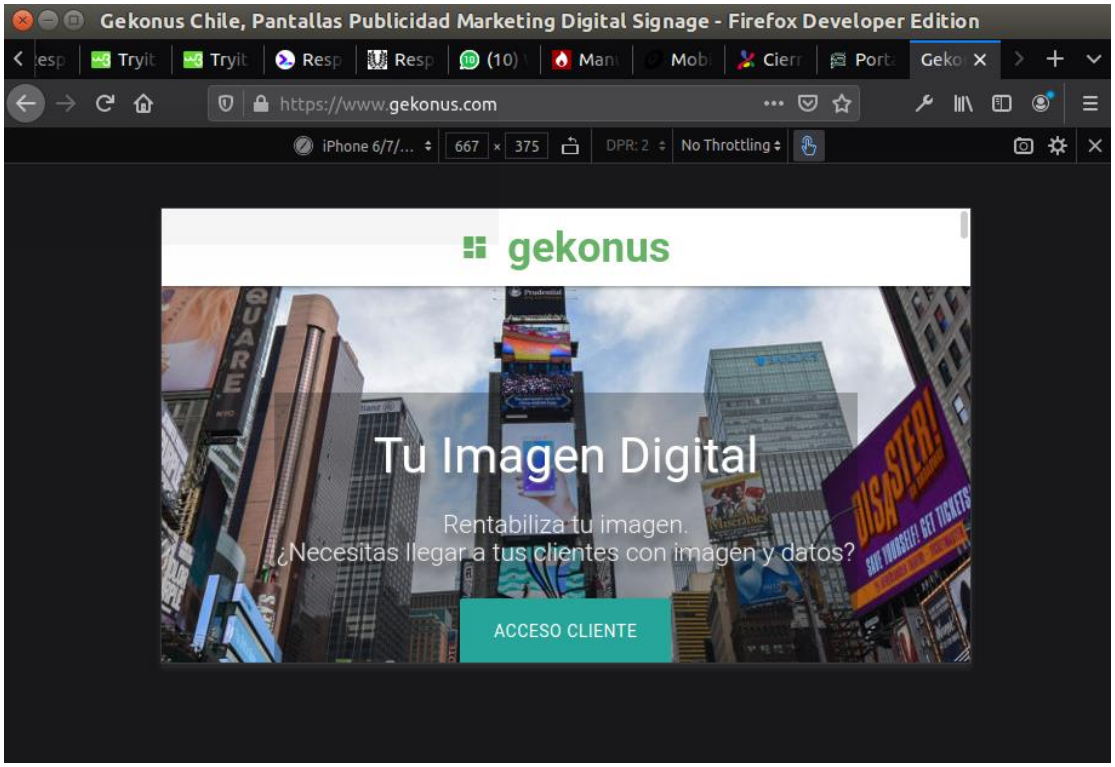
1.2.1.6. Cómo probar los distintos dispositivos

Existen diversas herramientas para simular cómo se apreciará nuestro sitio web en diversos dispositivos de dispositivos. En primera instancia, las herramientas de desarrollador del navegador, en nuestro caso **Firefox Developer**, en su **Modo de Diseño Responsivo**, podrá simular diversos dispositivos y orientaciones como se muestra a continuación:

Vista iPhone 6/7/8 iOS 11 - Orientación Vertical



Vista iPhone 6/7/8 iOS 11 - Orientación Horizontal



Como la variedad y aparición de nuevos dispositivos es una realidad cambiante a diario, es útil utilizar plataformas online que están sufriendo actualizaciones permanentes para entregar simulaciones más completas de la infinidad de dispositivos del mercado. Entre ellas se cuentan:

Resizer: https://material.io/resources/resizer/

Responsinator: http://www.responsinator.com/

1.2.2. Referencias

[1] Responsive Web Design, Ethan Marcotte, 2010.
https://alistapart.com/article/responsive-web-design/

[2] Responsive web design basics, How to choose breakpoints.
https://web.dev/responsive-web-design-basics/#breakpoints

[3] Manual de HTML: Responsive Web Design
http://www.estrellateyarde.org/manual-de-html/manual-de-html-responsive-web-design-diseno-web-adaptativo

[4] HTML Responsive Web Design
https://www.w3schools.com/html/html_responsive.asp

[5] CSS Units
https://www.w3schools.com/cssref/css_units.asp

[6] Manual de Responsive Web Design
http://www.desarrolloweb.com/manuales/responsive-web-design.html

[7] Responsive Web Design
https://www.w3schools.com/css/css_rwd_intro.asp

[8] Benjamin Lagrone, HTML5 and CSS3 Responsive Web Design Cookbook, 2013.

[9] Ben Frain, Responsive Web Design with HTML5 and CSS3, 2015, Second Edition.