

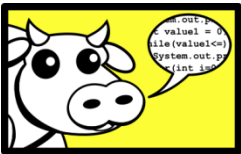
c. Introducción al RWD

Código Inicial para un sitio RWD

Como se mencionó antes, el *Responsive Web Design* implica un diseño que automáticamente se adapte al dispositivo que se esté utilizando para ver la página. Para ello, la página debe ser capaz de detectar las dimensiones del *viewport*, y los elementos de la página deben redimensionarse u ocultarse para que la misma se visualice siempre correctamente.

Configurando el *viewport*

En ese sentido, el primer paso es configurar la página para que evalúe correctamente el *viewport*. Para ello es necesario agregar una línea de código a todas nuestras páginas mediante la siguiente etiqueta **meta** en la sección **head** de las mismas:



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

Mediante esta etiqueta en el **head**, el navegador tendrá los elementos necesarios como para manejar mejor el escalado de nuestra página en diferentes tamaños del *viewport*.



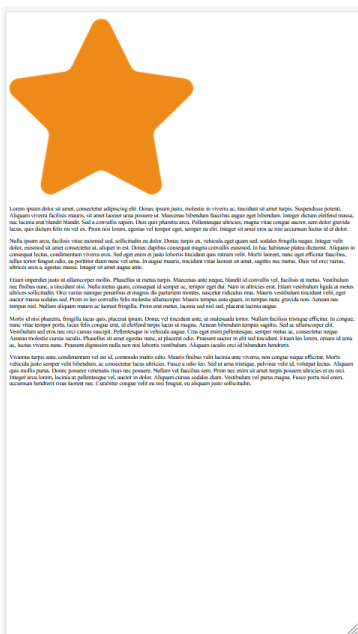
Atajos de Código en VSCode

Si estamos utilizando VSCode, y este está correctamente configurado, basta con ingresar el siguiente atajo de código y presionar “enter” al comenzar una página:

```
html:5
```

Al hacerlo VSCode generará un “esqueleto” de código básico para nuestra página que incluye el **meta** mencionado arriba.

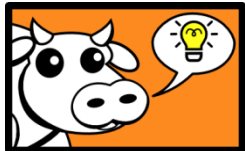
Como ejemplo, es recomendable que compares como cambia la visualización de las siguientes páginas:




Página **sin** viewport configurado
(<https://ejemplosinviewport.kurotori.repl.co/>)



Página **con** viewport configurado
(<https://ejemploconviewport.kurotori.repl.co/>)



Herramientas de Desarrollo: Modo de Diseño Adaptable

Tanto Mozilla Firefox como Google Chrome ofrecen un modo de visualización que emula el viewport de tablets y celulares. Para acceder a dicho modo se debe presionar el botón F12 (para mostrar el cuadro de herramientas de desarrollo, muy útiles para el desarrollo web) y dar clic en el ícono  que aparece en la parte superior.

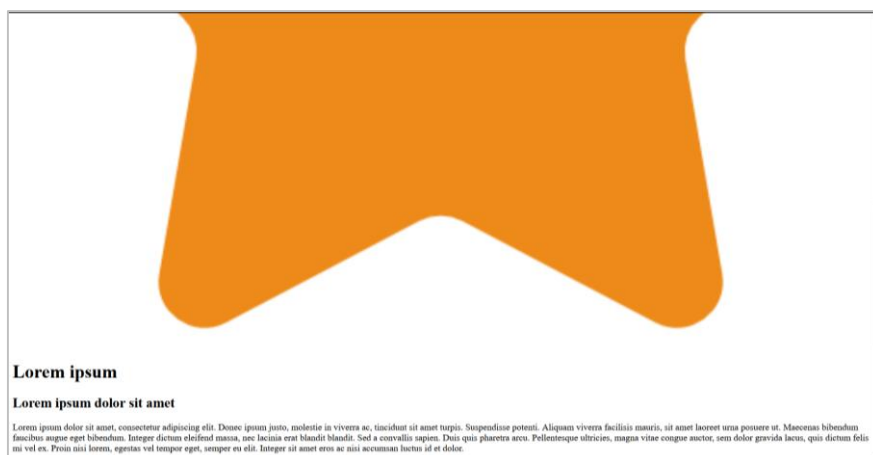
Utilizando esta funcionalidad es posible observar las diferencias mencionadas antes.

Imágenes Responsive: Configurando los tamaños máximos de los elementos

Si observamos el segundo ejemplo de más arriba, podemos ver que, a pesar de establecer el *viewport*, la imagen no se puede ver de forma correcta, ya que su ancho es mayor a lo que muestra la pantalla. En medios portátiles, los usuarios prefieren el desplazamiento vertical al horizontal, por lo que debemos evitar a toda costa que nuestra página exceda el ancho del *viewport* en estos dispositivos.



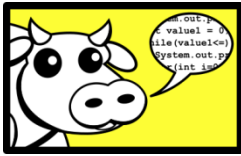
Para evitar esto debemos aplicar reglas al CSS que prevengan esto. Una forma de evitarlo sería estableciendo la propiedad CSS **width** de la etiqueta **img** en 100%, pero eso nos traerá un nuevo problema en las pantallas grandes, ya que la imagen se visualizará más grande que su tamaño original, perdiendo calidad:



Por una parte debemos evitar que las imágenes se muestren más grandes que el *viewport*, por otra parte, también debemos evitar que las mismas se vean demasiado grandes cuando el *viewport* es de mayor tamaño.

Esto se logra mediante la propiedad **max-width** de CSS. Esta propiedad establece el ancho máximo que puede tener un elemento. En este caso le aplicaremos esa propiedad a la etiqueta **img**, aunque en otros casos puede resultar más práctico crear una clase de CSS específica.

De esta manera obtenemos este código en el CSS de la página:



estilo.css

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

El resultado es el siguiente:

Viewport Móvil



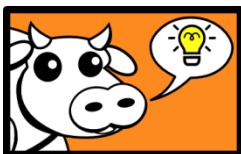
Viewport de PC



Fuentes Responsive: Texto que se ajusta al tamaño de la pantalla

Otro problema que podemos observar es que, en el caso del *viewport* de PC, el texto se ve demasiado pequeño en comparación con el tamaño de la pantalla. Eso se debe a que el texto se muestra con un tamaño de fuente fijo, debemos utilizar un tamaño de fuente relativo.

Para lograrlo esto, en vez de puntos o pixeles, podemos utilizar la unidad **vh**.



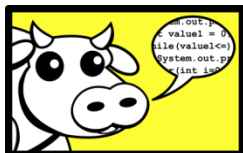
Unidades de medida Adaptables

Existen varias unidades de tamaño adaptables que podemos utilizar en nuestro diseño. A lo largo de este curso iremos conociendo varias y sus circunstancias de uso. Comencemos con las unidades basadas en el *viewport*:

Unidad	Abreviación	Tamaño relativo
Viewport width	vw	1 vw = 1% del ancho del <i>viewport</i>
Viewport height	vh	1 vh = 1% del alto del <i>viewport</i>
Viewport minimum	vmin	1 vmin = 1% del lado más chico del <i>viewport</i>
Viewport maximum	vmax	1 vmax = 1% del lado más grande del <i>viewport</i>

Utilizando unidades de tamaño relativas, podemos escalar la fuente de forma que el texto se visualice de forma proporcional al tamaño del *viewport* en cualquier dispositivo.

Por lo tanto le añadiremos las siguientes propiedades a las etiquetas **p**, **h1** y **h2** de nuestro ejemplo:



estilo.css

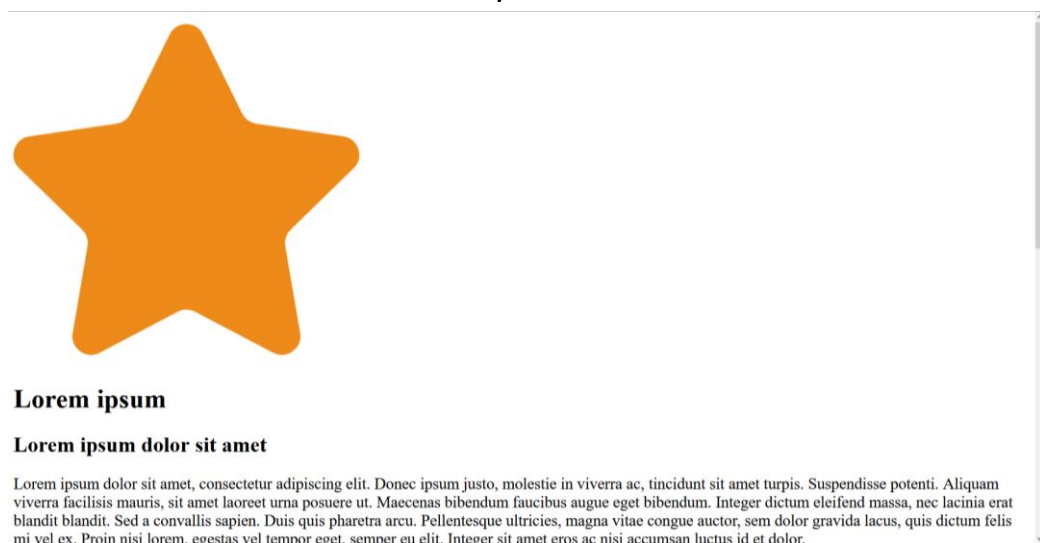
```
p {
    font-size: 3vh;
}
h1 {
    font-size: 5vh;
}
h2 {
    font-size: 4vh;
}
```

Esto solucionará nuestro problema de visualización de la fuente para los dos casos:

Viewport Móvil



Viewport de PC



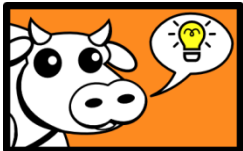
En otros casos, sin embargo, puede suceder que las diferencias de dimensiones de un viewport a otro sean muy grandes como para que una sola solución abarque correctamente todas las posibilidades. Esto es sobre todo el caso de diseños más complejos.

Es por eso que el siguiente paso es agregar a nuestra página una **media query**.

Media Query: Establecer propiedades para diferentes casos

Para eliminar los problemas que plantea el uso de un solo set de propiedades para todas las situaciones, lo que se suele aplicar al diseño es **diferentes sets de propiedades para los mismos elementos**, los cuales se aplican de acuerdo a las dimensiones del *viewport*.

Esto se logra definiendo *puntos de quiebre* en el código CSS indicando *las circunstancias* de aplicación de las propiedades.



Media Queries

Son bloques de propiedades CSS que se aplican de acuerdo a una serie de condiciones. Para establecer un bloque **media query** se utiliza el siguiente código:

```
@media screen and (condición) {
    ...
}
```

Dentro del bloque del **media query** incluiremos **todas las propiedades** que deben tener los elementos de nuestra página **para una determinada condición**.

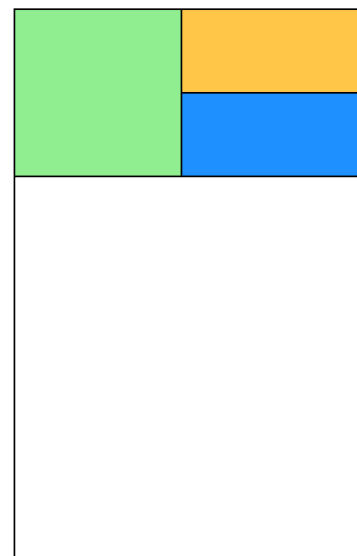
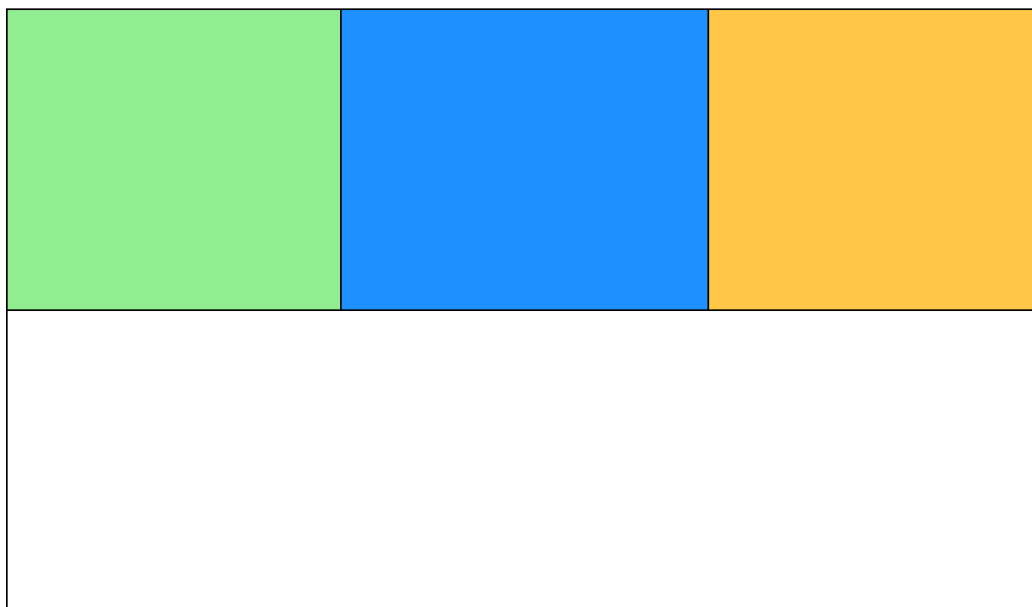
Las condiciones se establecen (entro otros casos) para **determinadas dimensiones de viewport**, las cuales pueden especificarse de varias maneras:

Condición	Significado
max-width: __px	Las condiciones se aplican cuando el viewport tiene como máximo este ancho en pixeles.
min-width: __px	Las condiciones se aplican cuando el viewport tiene como mínimo este ancho en pixeles.
max-height: __px	Las condiciones se aplican cuando el viewport tiene como máximo este alto en pixeles.
min-height: __px	Las condiciones se aplican cuando el viewport tiene como mínimo este ancho en pixeles.

Se puede consultar una lista completa de condiciones en:

https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

Por ejemplo, consideremos la siguiente situación. Se desea diseñar un sitio que presente dos *layouts* diferentes, dependiendo de si el *viewport* cambia de PC a móvil.



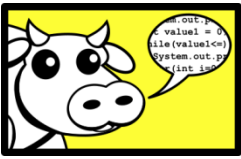
El enfoque del diseño hoy en día es *“mobile first”*, o sea, **primero diseñamos el sitio para las plataformas móviles**, y luego para PC.

En cuanto a código, podemos ver que, aplicando los criterios que vimos antes, tenemos en principio cuatro DIVs:

- El área superior que contiene los elementos coloreados.

- Los tres elementos coloreados.

Transformemos esto en código HTML:



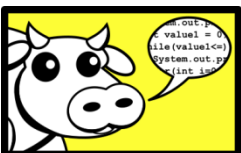
index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="estilo.css">
  <title>Document</title>
</head>
<body>
  <div id="areaSuperior">
    <div id="verde">
    </div>
    <div id="celeste" class="secundario">
    </div>
    <div id="naranja" class="secundario">
    </div>
  </div>
</body>
</html>
```

Los elementos **celeste** y **naranja** van a tener muchas características en común (dimensiones, posicionamiento, etc.), por lo que les asignamos una clase para evitar duplicar código.

Ahora, le aplicaremos CSS, pero **teniendo en cuenta el *layout* para el *viewport* móvil**. Según este *layout*, el área superior parece ocupar una tercera parte de la altura del *viewport*, por lo que le asignamos un 30%. El elemento **verde** ocupa toda la altura del área superior y aproximadamente la mitad de su ancho. Como su posición es contra la esquina superior izquierda, no nos preocupamos por su posicionamiento, pero los otros elementos deben **fluir por su derecha**, por lo que le aplicamos **flotabilidad hacia la izquierda**. Los elementos **naranja** y **celeste**, por su parte, deben fluir hacia la esquina superior derecha de la página, por lo que les aplicamos **flotabilidad hacia la derecha**.

El código resultante es este:



estilo.css

```
html,body{
  width: 100%;
  height: 100%;
  margin: 0px;
}

#areaSuperior{
  width: 100%;
  height: 30%;
}
```

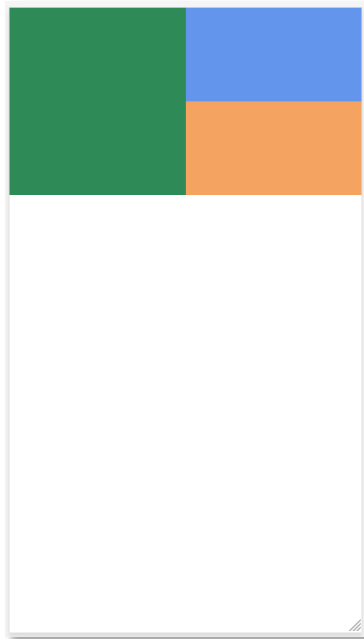
```
#verde{
    height: 100%;
    width: 50%;
    background-color: seagreen;
    float: left;
}

#naranja{
    background-color: sandybrown;
}

#celeste{
    background-color: cornflowerblue;
}

.secundario{
    float: right;
    height: 50%;
    width: 50%;
}
```

Y el resultado es este, (visto en modo de diseño adaptable):



Ahora haremos el mismo análisis, pero con el *layout* para *viewports* de PC, y le agregaremos al CSS el punto de quiebre correspondiente. En general podemos considerar que ese punto de quiebre está en unos 768px, por lo que nuestro código para viewports de PC estará en un bloque **media query** que identifique a pantallas de 768px de ancho o mayores. Conviene siempre añadir un comentario que aclare el objetivo del bloque.



estilo.css (continuación)

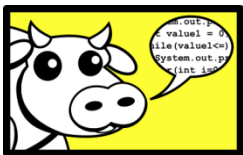
```
@media only screen and (min-width: 768px) {
    /* Estilo Para PC */
}
```

Las principales diferencias que encontramos el *layout* para *viewport* de PC son:

- El área superior ocupa el 50% del *viewport*.
- Los tres elementos ocupan el 100% de la altura del área superior, y cada uno ocupa una tercera parte del ancho (que resulta en un 33.33%)
- Cada elemento fluye por la derecha del anterior, por lo que todos deben tener flotabilidad hacia la izquierda.

Todas las otras propiedades (el color de fondo de cada elemento, por ejemplo) **se mantienen**. Por lo que **no es necesario declarar esas propiedades de nuevo**.

El bloque queda de esta manera:



estilo.css (continuación)

```
@media only screen and (min-width: 768px) {  
  /* Estilo Para PC */  
  #areaSuperior{  
    width: 100%;  
    height: 50% ;  
  }  
  
  #verde{  
    height: 100%;  
    width: 33.33%;  
    float: left;  
  }  
  
  .secundario{  
    float: left;  
    height: 100%;  
    width: 33.33%;  
  }  
}
```

Y el resultado en pantalla es el siguiente:



(Puedes ver este ejemplo en: <https://ejemplomediaquerymobilefirst.kurotori.repl.co/>)

En conclusión: obtuvimos una página cuyos elementos se comportan de forma diferente ante diferentes dimensiones del viewport, adaptándose al mismo, gracias a nuestro bloque **media query**.

Notemos, además, que al diseñar siguiendo el criterio *mobile first*, nos ahorramos trabajo, ya que no debemos considerar nuestra página para diferentes medidas de pantalla de dispositivo móvil, sino que solo para pantallas de PC, que son mucho más estandarizadas en sus dimensiones.