Layout

Hasta ahora hemos trabajado en la versión **mobile** de nuestro ejemplo.
Si vamos a una resolución mayor, vemos que nuestro layout es elástico.

Layout

Haciendo uso del CSS vamos a introducir cambios y organizar nuestro contenidos en columnas que se visualizarán en base a la resolución.

"Mobile first"
Comenzamos muy bien pensando nuestro diseño para celulares.

Para resoluciones mayores tendremos que hacer algunos cambios.

Para llevar adelante estos cambios en base a la resolución de pantalla, usamos:

?

Media Queries

Son consultas a los dispositivos.
Acciones que modifican y reacomodan contenidos en base a lo que le ordenemos hacer con **CSS**.S

Si a nuestro diseño lo vemos en nuestros celulares en este momento veremos que podemos aplicarle zoom, una propiedad que demuestra que todavía nuestro ejemplo no es RWD.

Desde el celu:
www.facundoabbiatti.com.ar/rwd/01.html

Media Queries

Viewport
Es la parte visible de nuestro navegador. Es la zona en la cual podemos aplicar nuestros estilos.
Es una etiqueta meta. Y bloqueará el uso del zoom.

Veamos estructura del viewport.

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

En el celular:
www.facundoabbiatti.com.ar/rwd/02.html

Media Queries

Declaración de las media queries

```
@media screen and (min-width:800px){
}
```

Estamos declarando que dependiendo de la salida de pantalla del dispositivo, cuando su mínimo ancho sea de 800px, aplique lo que dispongamos.

@media print

Muy útil para cuando queremos por ejemplo que nuestro diseño no consuma mucha tinta. Podemos aplicarle determinados valores que al imprimir la web, sólo determinados elementos se impriman.

Media Queries

Llevemos nuestros articles a 2 columnas

```
@media screen and (min-width:800px){

    body
    {
        background: red;
    }
    #contenido .item
    {
        display: inline-block; /* Para romper */
        width: 49%;
    }
```

De esta forma no divide en 2 columnas porque siempre inline blok da un pequeño margen.

Para evitar tener que hacer esto y "ensuciar", pensemos una mejor solución.

Media Queries

Para no tener que modificar el ancho a 49%

```
#contenido .item
{
    float: left;
    width: 50%;
}
```

Como ya vimos, si aplicamos un float, a nuestra caja contenedora tenemos que aplicarle overflor hidden para que no se salga ningún elemento.

Para controlar nosotros y dejar un espacio entre columnas, podríamos probar algo así:

```
#contenido .item
{
    float: left;
    width: 49%;
    margin-right: 1%,
}
```

Media Queries

Visualmente la columna derecha no queda centrada. Entonces podemos alinear al centro sólo una de los 2 columnas resultantes.
Para ellos utilizamos algo que ya vimos, pseudo selector. Entonces primero, eliminamos el margin que pusimos:

```
#contenido .item{
    float: left;
    width: 49%;
}

#contenido .item:nth-child(odd){ /* Impares */
    margin-right: 2%;
}
```

Como nuestro layout tiene 2 columnas, siempre tomará la primera de la izquierda como impar.

Media Queries

Basta por hoy...

Layout

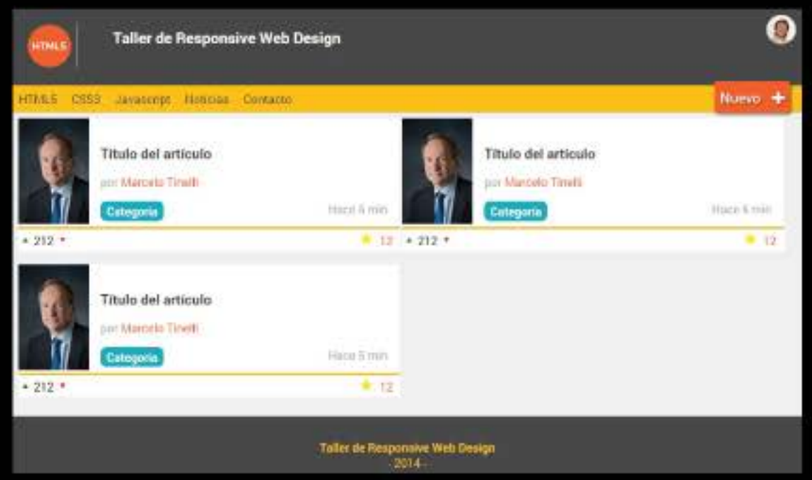
@media screen and (min-width:800px)

Para "limpiar" nuestra técnica volvamos al 50% y flotemos hacia la izquierda. Esto habilitará a que los article se ubiquen unos sobre otros.

```
#contenido .item {
    float: left;
    width: 50%;
}
```

Si flotamos algún elemento, ya habíamos visto que es necesario que a nuestro contenedor le aplique-mos overflow hidden para que no se "salgan".

```
#contenido {
    overflow: hidden;
}
```



Layout

@media screen and (min-width:800px)

Vamos a controlar un margen derecho.

```
#contenido .item {
    float: left;
    margin-right: 1%;
    width: 49%;
}
```

Ahora vamos a controlar la posición de los articles. Para ello necesitamos un pseudo selector de css.

Como vimos antes, tenemos **first-child** y **last-child** pero también existe **nth-child()** que selecciona pares e impares. (even, odd) Entonces seleccionamos los elementos impares de esta forma:

```
#contenido .item:nth-child(odd) {
    background: red;
}
```

Layout

@media screen and (min-width:800px)

Controlamos el margen derecho sólo de los elementos de la izquierda.

```
#contenido .item:nth-child(odd) {
    margin-right: 2%;
}
```

También eliminamos el margen que habíamos puesto a:

```
#contenido .item
    {
        float: left;
        width: 49%;
    }
```

Como tenemos nuestro diseño a 2 columnas, siempre el impar va a estar a la izquierda.

Video

http://www.w3schools.com/html/html5_video.asp

Los navegadores que soportan webm soportan también ogg.
Ambos son 2 formatos "abiertos" de video.

Nuestro amigo IE, siemrpe da la nota y no soporta ninguno de esos... soporta el mp4

Por lo que debemos aclarar, mp4 y por lo menos uno de los otros 2 mencionados. Debemos tener nuestro video convertido a esos 2 formatos.

```
<video width="320" height="240" controls>
    <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
    Your browser does not support the video tag.
</video>
```

Video

¿Cómo hacemos para que este video se muestre únicamente para resoluciones mayores a 1000px? Primero, por defecto, a nuestro video le tenemos que decir que no se visualice, ya que con las media queries vamos a controlar eso.

```
video {
    display: none;
    }
```

Supongamos que es contenido secundario, entonces es hora de aplicar la etiqueta html semántica aside.

```
<aside>
    <video width="320" height="240" autoplay>
    <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
    Your browser does not support the video tag.
    </video>
</aside>
```

```
@media screen and (min-width:1000px){
    aside{
        display: block;
        float: right;
        margin: .5em;
    }
```

Animaciones

La próxima clase...