

- Responsive Web Design
 - Introducción
 - Conceptos básicos
 - Preparación previa
 - Diseño con porcentajes.
 - Fluid Layout
 - Las media queries
 - Ejemplos de media queries
 - ¿Cuántos breakpoints tenemos que definir?
 - Fluid y Media queries

RESPONSIVE WEB DESIGN

INTRODUCCIÓN

Se denomina Responsive Web Design (o RWD) a los diseños web que tienen la capacidad de adaptarse al tamaño y formato de la pantalla en la que se visualiza el contenido.

En [MediaQueri](#) puedes encontrar algunos ejemplos de páginas que utilizan Responsive Web Design para tener clara la idea.

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. Que vamos a resumir aquí:

- La diferencia entre diseño responsivo y diseño adaptativo. Un diseño **responsive** se adapta en todo momento a cada una de las dimensiones de la pantalla, mientras que un diseño adaptativo solo en determinados casos.

- Para trabajar correctamente en diseños **responsive** hay que tener en cuenta que debemos trabajar con **unidades relativas** e intentar evitar las unidades fijas o estáticas.
- Utilizar propiedades como `min-width o max-width`, donde definimos tamaños mínimos o máximos, para que los elementos de nuestra página puedan ampliar o reducirse según sea necesario dependiendo de la pantalla del dispositivo utilizado.
- Mantener el flujo de los elementos cuando cambian de tamaño y evitar que estos se solapen unos con otros.

En las siguientes secciones se desarrollan algunos de estos conceptos y se ampliarán.

Preparación previa

Es importante conocer los formatos de pantalla más comunes con los cuales nos vamos a encontrar. Podemos consultar páginas como [MyDevices](#), la cuál tiene un apartado de comparación de dispositivos, donde se nos muestra un listado de dispositivos categorizados en smartphones, tablets u otros dispositivos con las características de cada uno: dimensiones de ancho, alto, radio de píxels, etc...

Otro punto importante, es el estudio de nuestro público objetivo. Que hicimos en el primer tema, que nos dan la respuesta a la preguntas: "¿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?".

Por último y no menos importante es la estrategia de diseño que vamos a optar, con la información que hemos recogido anteriormente. Tenemos dos:

- **Mobile first** Primero nos enfocamos en dispositivos móviles y luego pensamos en otros.
- **Desktop first** Primero nos enfocamos en dispositivos de escritorio, y luego pensamos en otros.

La estrategia Mobile-first es la que utilizan los diseñadores de sitios webs en las que su público objetivo es mayoritariamente usuario de móvil. Ejemplos como una web para comprar billetes de transporte, la web de un juego o aplicación móvil o una web para pedir cita en un restaurante podrían ser, a priori, una buena elección para utilizar Mobile-first.

Esta estrategia hace que el desarrollo en escritorio sea muy sencillo, ya que se reduce a tener un diseño de móvil en escritorio e ir añadiendo nuevas secciones o partes para «completar» el diseño en resoluciones grandes.

Por otro lado, la estrategia Desktop-first suele interesar más a los diseñadores de sitios webs en las que el público objetivo son usuarios de escritorio. Por ejemplo, una página de una aplicación para PC/Mac o similares, podría ser una buena opción para la estrategia Desktop-first. En ella, hacemos justo lo contrario que en la anterior, lo primero que diseñamos es la versión de escritorio, y luego vamos descargando detalles o recolocando información hasta tener la versión para dispositivos móviles.

DISEÑO CON PORCENTAJES.

Una de los principios que tenemos que tener en cuenta es que todas las medidas de todos los elementos serán relativas. Dos tipos de medidas que utilizaremos son **rem** y **%**.

Para hacer la transformación de todas nuestras medidas desde px a por ejemplo em hay que utilizar la formula:

resultado=objetivo/contexto

Vamos a ver un ejemplo sobre el tamaño de las fuentes.

```
<h1>Hola mundo <a href="">Esto es un link</a></h1>
```

```
html {
  font-size: 16px;
}
body {
  color: #352a25;
  font-family: Georgia, serif;
  font-size: 62.5%; /* 1em = 10px */
}
h1 {
  font-size: 3em; /* 30/10=3rem */
  font-weight: bold;
}
```

Este código css ya lo hemos utilizado para que **1em** sea igual a **10px**. El siguiente problema que tenemos es contextualizar el tamaño de la letra del elemento `a`, le queremos dar **14px**. Como sería lógico pensar la formula que se va aplicar es 14px/10px pero esto no funcionará y el resultado es el siguiente.

Hola mundo ESTO ES UN LINK

Porque el contexto del link no es 10px si no 30px. Dado que los caracteres de un elemento en **em** se calculará tomando como referencia su elemento padre. El código correcto sería el siguiente:

```
h1 a {
  font-size: 0.46666667em; /* 14px/30px */
  text-transform: uppercase;
  text-decoration: none;
  color: #6c564b;
}
```

Para superar el inconveniente indiscutible que supone la cascada a la hora de calcular los tamaños relativos de los caracteres, se utiliza el **rem**.

Fluid Layout

En este apartado vamos a utilizar porcentajes en nuestro layout.


Hay que tener en cuenta que el usuarios puede modificar el tamaño de la ventana o consultar la web dese un móvil o tableta con una pantalla más pequeña. Al realizar un **diseño fluido** todos los elementos y más el diseño tienen que distribuirse proporcionalmente en toda la pantalla.

Fluid layout se adapta a resoluciones de pantalla y a dispositivos, pero esto es su vez una desventaja cuando el ancho del dispositivo es muy grande y el contenido es escaso. También puede darse el caso contrario, que el ancho disponible del dispositivo sea demasiado pequeño y se haga uso de la propiedad `min-width` en el cual aparecerán las temidas barras horizontales que siempre deben evitarse.

Con el siguiente ejemplo vamos ilustrar la problemática de ajustar a porcentajes todos los valores en px. Vamos a empezar con un esquema con dimensiones fijas, en píxeles. A continuación, veremos cómo pasar de este diseño a uno fluido.


ENCABEZADO DE LA PÁGINA WEB

Encabezado del artículo 1



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Tenetur illum tempora eaque animi culpa. Accusantium maxime illo, id minus perferendis a ea vitae aut excepturi fugit impedit, sit voluptatibus amet.

Encabezado del artículo 2



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repudiandae dignissimos iusto sunt eveniet, consequatur. Aliquam ratione autem incidunt, nihil asperiores magni modi reprehenderit at eaque magnam! Perspiciatis mollitia praesentium ex.

*At nos hinc posthac
Gallia est omnis divisa in
partes tres, quarum.
Curabitur blandit tempus
ardua ridiculus sed magna.
Prima luce, cum quibus
mons aliud consensu ab eo.
Plura mihi bona sunt,
inclinat, amari petere
vellent. Etiam habebis sem
dicantur magna mollis
euismod.*

Plura mihi bona sunt

*Curabitur blandit tempus
ardua ridiculus sed magna.
Phasellus laoreet lorem vel
dolor tempus vehicula.
Fictum, deserunt mollit
anim laborum astutumque!
Quid securi etiam tamquam
eu fugiat nulla pariat.*

Ejemplo sacado de Bruno Sébarte de video2brain

Crea la anterior pagina web con el siguiente código HTML:

```

<div id="container">
  <header>
    <h1>Lorem ipsum dolor...</h1>
  </header>
  <main>
    <article>
      <h2>Prima luce cum quibus</h2>
      <figure></figure>
      <p>Magna pars studiorum...</p>
      <p>Etiam habebis sem dicantur magna...</p>
    </article>
    <article>
      <h2>Mons aliud consensu</h2>
      <figure></figure>
      <p>Curabitur blandit tempus ardua...</p>
    </article>
    <aside>
      <h4>At nos hinc posthac</h4>
      <p>Gallia est omnis divisa in partes tres...</p>
      <h4>Plura mihi bona sunt</h4>
      <p>Curabitur blandit tempus ardua...</p>
    </aside>
  </main>
</div>

```

```

    </aside>
  </main>
  <footer>
    <p>Curabitur est gravida et libero vitae dictum.</p>
  </footer>
</div>

```

El contenedor principal contenedor tiene 960 píxeles de ancho y está centrado en la pantalla.

- Las dos cajas `article` tienen 360 píxeles de ancho para el contenido, 10 píxeles de relleno, es decir, un ancho en pantalla de 380 píxeles.
- La columna lateral derecha, `aside`, tiene 180 píxeles de ancho para el contenido y 10 píxeles de relleno, lo que nos da un ancho en pantalla de 200 píxeles.
- Las imágenes son más grande que su contenedor, por lo que vamos a imponerlos un ancho fijo de 360 píxeles.

Lo primero es el contenedor general `container`. En nuestro diseño fijo, tenía 960 píxeles de ancho y estaba centrado en la ventana del navegador. Para ello no tenemos más que la referencia de la pantalla ya que el padre es el body. Podemos darle el siguiente código css.

```

#container {
  width: 90%;
  max-width: 1200px;
  margin: 2% auto;
}

```

Que sentido dar ancho fijo máximo pues, para que la página llegue a un ancho máximo aunque el usuario cambie el tamaño de la ventana del navegador más allá de ese valor.

En los otros elementos `<header>`, `<main>` y `<footer>`, no es necesario indicar valores, ya que heredan los de su elemento padre, el elemento `<container>` .:

- El elemento padre de `<article>` es `<main>`, cuyo ancho viene determinado por su propio padre, `<container>`, que tiene un ancho fijo de 960 píxeles. Para `article` $(359 \times 100) / 960 = 37,3958333333\%$
- Para el `padding` del elemento article. En el diseño fijo era de 10 píxeles, en porcentaje es $(10 \times 100) / 960 = 1,0416666667\%$.
- Para el `aside` $(180 \times 100) / 960 = 18,75\%$.

¿hay que dejar todo esos decimales? Y la respuesta es: ¡sí! Si quitamos decimales, corremos el riesgo de que la suma de todos estos valores simplificados no dé un total del 100%.

Y ahora que pasa con las imágenes que son más grandes que todo el ancho. Para tener una imágenes fluidas tenemos que establecer dónde está ubicadas. El elemento ``, colocado a su vez en `<figure>`, cuyo elemento padre es `<article>`. En el diseño fijo, el elemento `<figure>` puede ocupar 360 píxeles de ancho `(width=360px;)` en el elemento `<article>`. Este último tiene 380 píxeles de ancho en pantalla, lo que nos da el contexto. La fórmula será la siguiente: **$(360 \times 100) / 380 = 94.7368421052$** .

Ahora habrá que establecer un ancho máximo para las imágenes.

```
h1,
h2,
h4,
p,
figure,
img {
  margin: 0;
} /* añade esto para poder poder controlar las imagenes.*/
figure {
  width: 94.7368421052%;
}
figure img {
  max-width: 100%;
}
```

Si en ejemplo anterior utilizamos el inspector para modificar el tamaño de la pantalla nos damos cuenta en que hay situaciones en las que determinados aspectos o componentes visuales deben aparecer en un tipo de dispositivos, o deben existir ciertas diferencias.

Para ello, utilizaremos un concepto denominado media queries, con los que podemos hacer esas excepciones para que sólo se apliquen a un tipo de diseño concreto.

LAS MEDIA QUERIES

El **módulo Media Queries** propone determinar que tipo de medio, mediante criterios precisos y la posibilidad de combinar estos criterios. El valor que devuelve un Media Query es de tipo booleano: verdadero o falso.

Por ejemplo:

```
@media screen and (*condición*) {  
  /* reglas CSS */  
  /* reglas CSS */  
}  
  
@media screen and not (*condición*) {  
  /* reglas CSS */  
  /* reglas CSS */  
}
```

En este caso solo en pantallas que cumplan las condiciones que aparecen en paréntesis. También se pueden indicar reglas `@media` negadas mediante la palabra clave `not`, que aplicará CSS siempre y cuando no se cumpla una determinada condición. Además, pueden separarse por comas varias condiciones de media queries

Existen los siguientes tipos de medios:

| Tipo de medio | Significado |
|---------------------|---|
| <code>screen</code> | Monitores o pantallas de ordenador. Es el más común. |
| <code>print</code> | Documentos de medios impresos o pantallas de previsualización de impresión. |
| <code>speech</code> | Lectores de texto para invidentes (Antes aural, el cuál ya está obsoleto). |
| <code>all</code> | Todos los dispositivos o medios. El que se utiliza por defecto. |

Con el siguiente fragmento de código HTML estamos indicando que el nuevo ancho de la pantalla es el ancho del dispositivo, por lo que el aspecto del viewport se va a adaptar consecuentemente:

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


Ejemplos de media queries

El código sería el siguiente:

```
@media screen and (max-width: 640px) {  
  .menu {  
    background: blue;  
  }  
}  
  
@media screen and (min-width: 640px) and (max-width: 1280px) and  
(orientation: landscape) {  
  .menu {  
    background: red;  
  }  
}  
  
@media screen and (min-width: 1280px) {  
  .menu {  
    background: green;  
  }  
}
```

El ejemplo anterior muestra un elemento (con clase menu) con un color de fondo concreto, dependiendo del tipo de medio con el que se visualice la página:

- Azul para resoluciones menores a 640 píxeles de ancho (móviles).
- Rojo para resoluciones entre 640 píxeles y 1280 píxeles de ancho (tablets) y la orientación es landscape
- Verde para resoluciones mayores a 1280 píxeles (desktop).

¿Cuántos breakpoints tenemos que definir?

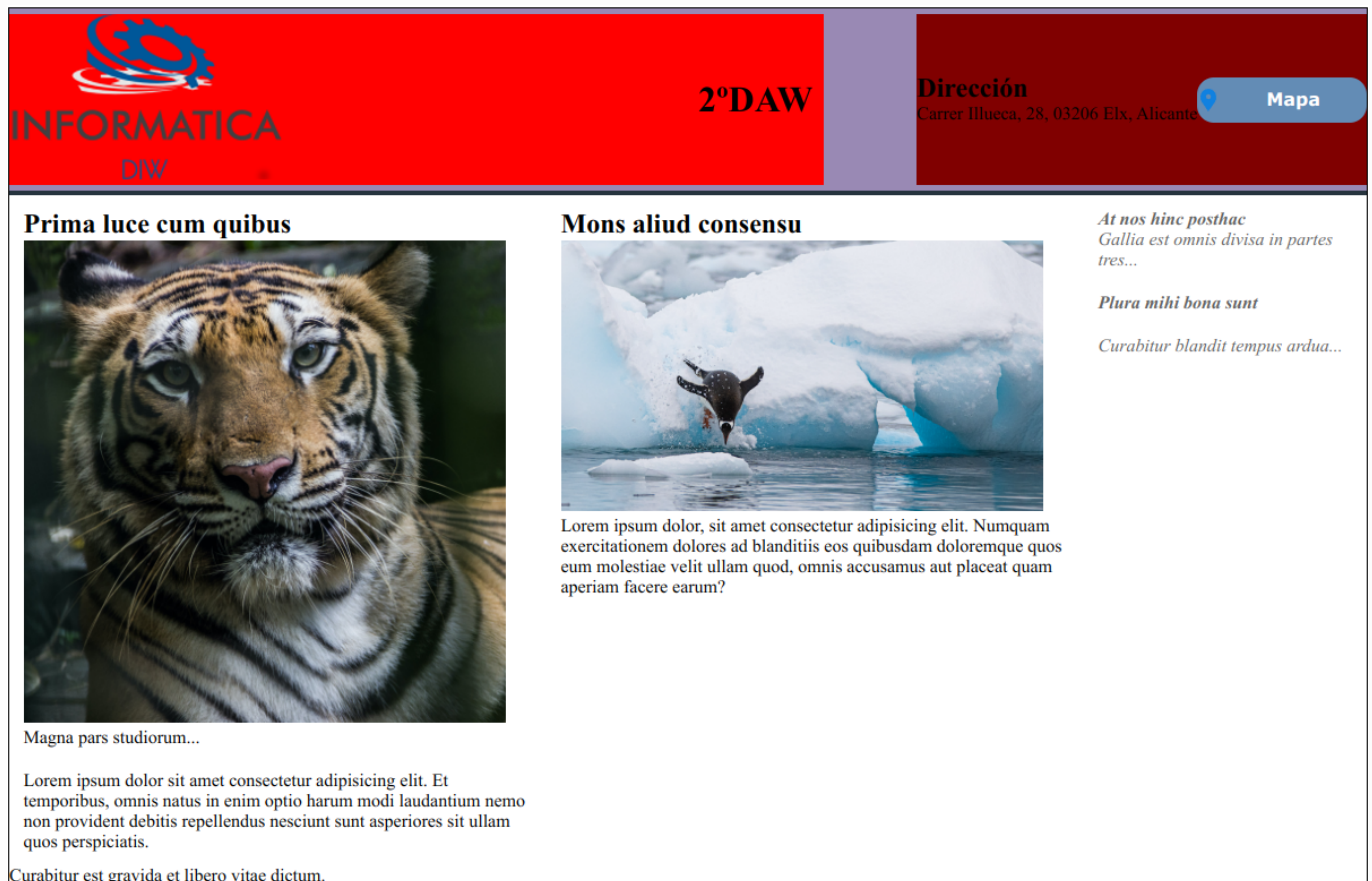
Como a partir de ahora los diseños son fluidos y si tenemos el siguiente ejemplo.

```
@media screen and (max-width: 320px) {  
}  
@media screen and (max-width: 480px) {  
}
```

Sólo habrá que definir el más grande (480px) ya que el diseño (estructura) se va a escalar al tamaño de la vista porque a partir de 480px se verá de la misma forma que con 320px.

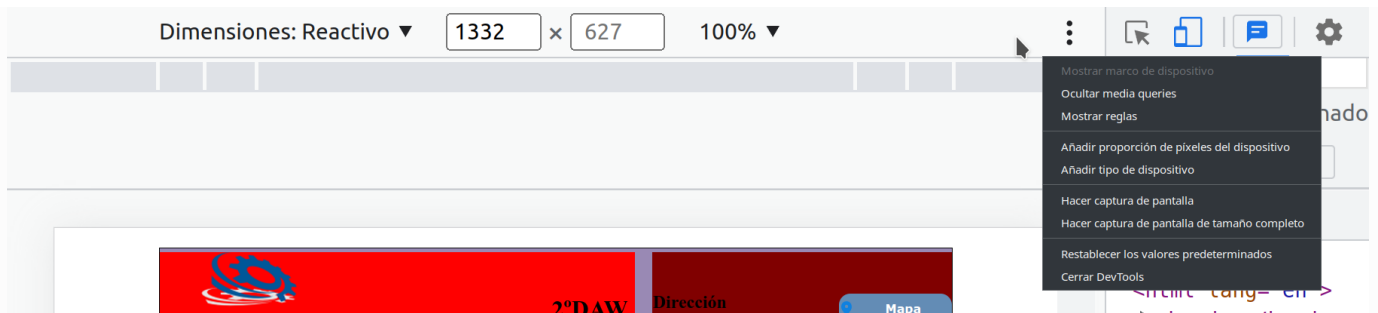
Fluid y Media queries

Ahora vamos a modificar el ejemplo que hemos trabajado en el apartado de Fluid Layout para que se muestre de la siguiente manera:



Como vemos , hemos añadido en el header dos elementos uno, el logo con un título (`logo-text`) y otro con la dirección con un botón (`localización`). Entre ellos he dejado un espacio entre ellos.

Si abrimos el inspector con el navegador en la parte superior derecha tenemos **ocultar media queries** esta opción nos ocultará o dispondrá en la barra superior zonas donde hemos establecido los puntos de ruptura.

**Nota:**

Esta imagen pertenece al inspector de Google Chrome.

Aquí os dejo algo del código de css que no tiene que ver con la estructura. Ya que la estructura la tendréis que realizar vosotros. Esta corresponde con la **Actividad 1**.

```
#boton_mapa {
  text-align: center;
  font-family: Verdana, Helvetica;
  font-weight: bold;
  color: white;
  background: #638cb5;
  border-radius: 15px;
  width: 150px;

  padding: 10px 0px;
  float: left;
}

#boton_mapa::before {
  content: "";
  display: block;
  background: url("../img/placeholder.svg") no-repeat;
  width: 20px;
  height: 20px;
  float: left;
  padding: 0;
}
```

Para saber que breakpoints debemos introducir hay que analizar la estructura de nuestra web y de cada uno los elementos.

Por lo que , el contenido va a definir los breakpoints. Si hacemos más pequeña la pantalla de nuestro navegador y teniendo en cuenta las dimensiones de los objetos podemos ir definiendo todos aquellos elementos que tenemos que variar.

Como podemos observar, llega un momento que a 1284px que la parte de `localización` y `logo-text` se juntan demasiado y hay un punto que los dos bloques se superponen y no caben.



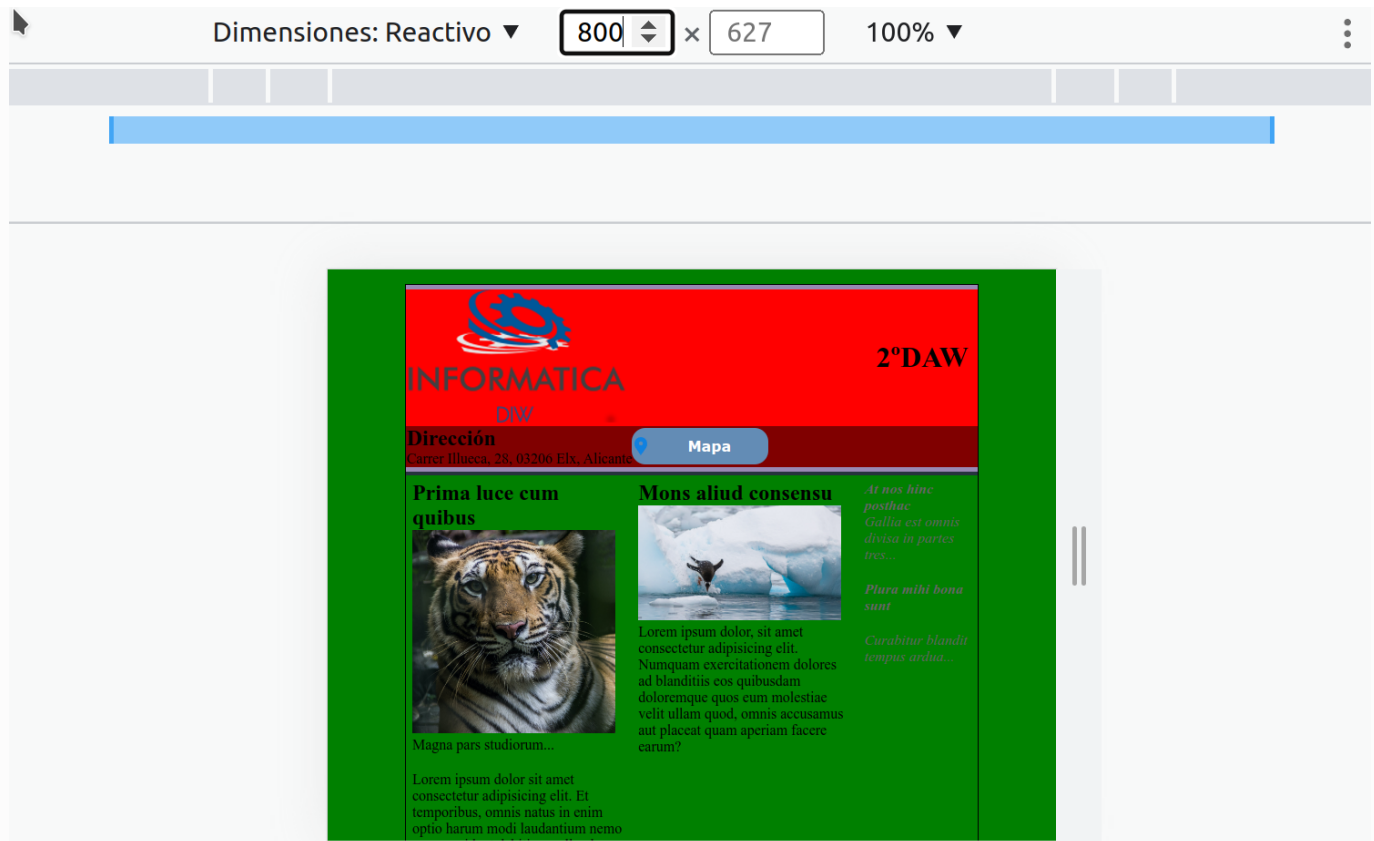
Ahora hemos establecido un breakpoint para ese punto, de tal forma que se nos muestre de la siguiente manera:



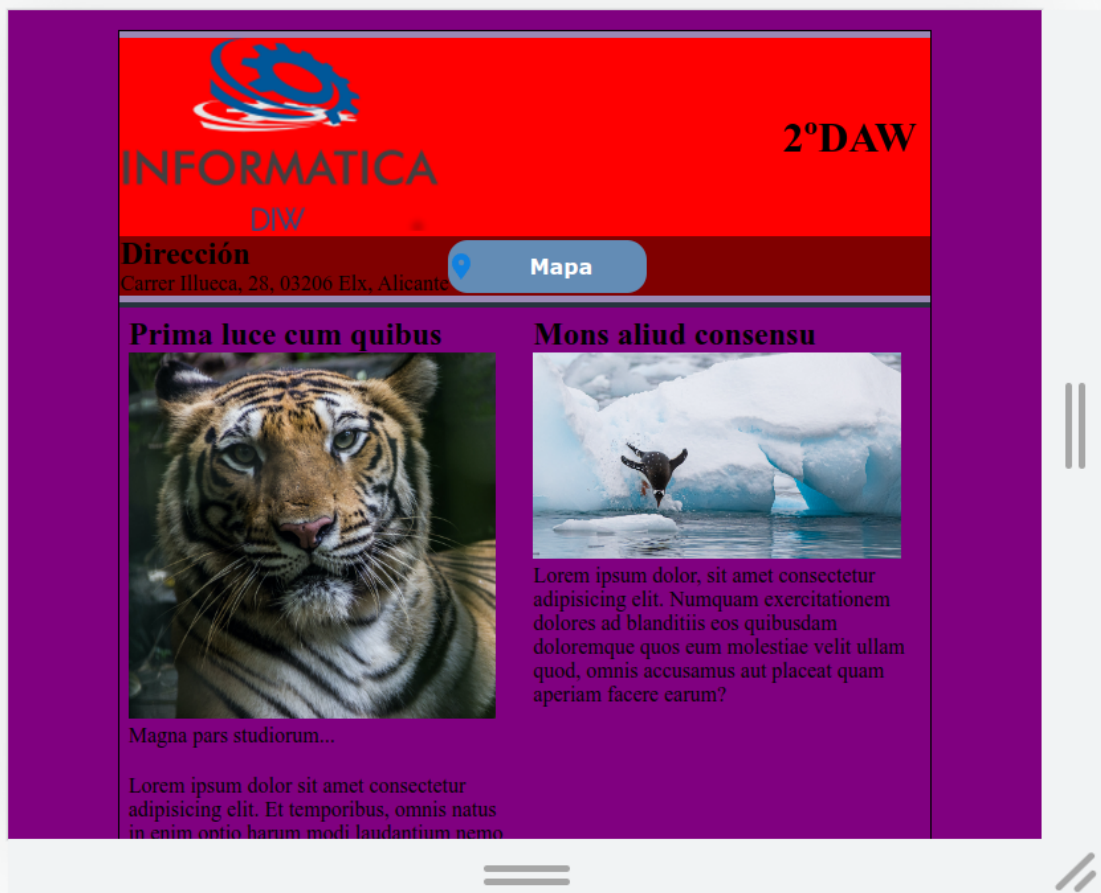
Como podemos ver con esta dimensión no afecta a los artículos solo al header. En esta solución `logo-text` se ha colocado arriba y `localización` se ha colocado debajo intentando cubrir todo

Una buena práctica cuando estemos modificando el tamaño de la ventana o cuando queramos identificar tamaños es poner bordes y colores de fondo distintos.

La siguiente imagen pertenece a un tamaño de 800px. Como podemos observar los artículos empiezan a ser ilegibles. Hay que dar una respuesta modificando el diseño. La solución por la que optado es la de hacer desaparecer el aside y los artículos dejarlos a 50%. El resultado es el siguiente.



Y la solución es la siguiente:



Ahora el próximo breakpoint se establece a los 580px cuando los artículos se hacen ilegibles y lo más óptimo es ponerlos uno encima de otros. Además es conveniente que se modifique ya el encabezado para utilizar esta adaptación para los móviles.

Y nos da como resultado:

**Nota:**

Este ejercicio lo he realizado con flex. Tener en cuenta que la propiedad `width` siempre va ser la que determine la renderización de los objetos.

Hay más mejoras con respecto al ejercicio pero estas he considerado que son las más importantes.

Con respecto a las imágenes, videos y cualquier objeto embebido. Deberemos guardar la imagen lo más grande posible y utilizar la propiedad `max-width: 100%;` controlando el tamaño con elemento padre que en nuestro caso es figure.

No olvidemos que utilizamos hojas de estilo en cascada, Cascading Style Sheet. Lo que implica que los estilos colocados al principio del archivo CSS se cargan primero y que a continuación pueden ser reemplazados por estilos colocados más abajo en el archivo.