



ESTHER CID

RESPONSIVE WEB DESIGN

Diseño de interfaces
Web

2º DAW



ÍNDICE

¿QUÉ ES RESPONSIVE WEB DESIGN?	3
CONSEJOS PARA CONSEGUIR UN DISEÑO WEB DE FORMA MAS SENCILLA.....	3
LOS TRES PILARES DE OPTIMIAZACIÓN DEL DISEÑO RESPONSIVE	4
PASOS PARA DISEÑAR UN SITIO WEB RESPONSIVE	4
DOS TÉCNICAS PARA RESOLVER PROBLEMAS.....	7
LA FILOSOFÍA MOBILE FIRST.....	7
DISEÑO	7
BREAKPOINT ORIENTADO AL DISEÑO RESPONSIVE	8
¿Dónde colocar un breakpoint?	9
UNIADES DE MEDIDA EN CSS	12
ESTILOS CSS BÁSICOS PARA RESPONSIVE WEB DESIGN.....	13
LA ETIQUETA VIEWPORT	13
EL ELEMENTO PICTURE	14
EL ELEMENTO SOURCE	14
DETECTAR MEDIA QUERIES CON JAVASCRIPT	15
EL MÉTODO WINDOW MATCHMEDIA	15

¿QUÉ ES RESPONSIVE WEB DESIGN?

Cuando hablamos de diseño responsive nos referimos o sea la forma en la que las páginas web se adaptan a los diferentes medios, no es lo mismo ver un diseño en una pantalla de ordenador de sobremesa, que en un teléfono móvil.

Hace unos años el único medio a través del cual podríamos consumir páginas web era un ordenador, normalmente de sobremesa, pero en la actualidad gran parte del consumo aparece desde terminales móviles que tienen un tamaño y una resolución de pantalla muy diferentes a las de un ordenador. También han entrado en el mercado nuevos dispositivos como tablets que tienen otra resolución diferente.

En la actualidad gracias a la implantación de CSS3 crear este tipo de adaptabilidad es muy sencillo, también tenemos a nuestra disposición plantillas con el responsive ya hecho.

Dentro de CSS3 nos referimos a las media queries, que son unas construcciones del lenguaje CSS que permiten definir estilos condicionales que solo se aplican cuando suceden determinadas situaciones, o a atributos tan simples como “max-width” o “min-width”.

CONSEJOS PARA CONSEGUIR UN DISEÑO WEB DE FORMA MAS SENCILLA

1. El primer Consejo es no utilizar estilos “inline”, este tipo de estilos son los que se colocan en pedidos dentro del HTML, utilizar estos estilos nos encorseta a que si hay algún error dentro del CSS o en la página tengamos problemas a la hora de mirar de qué se trata.
2. No es necesario que la web se vea exactamente igual en todos los dispositivos en los que vaya a abrirse, lo que tenemos que conseguir es que se lea correctamente el usuarios puedan consumir el contenido.
3. No hay que diseñar para una plataforma, si lo hacemos de esta manera cuando la plataforma más utilizada sea otra tendremos que volver a optimizar, la estrategia del diseño es no centrarse en eso y realizar un diseño general.
4. Debemos utilizar javascript solo cuando sea necesario, es decir, si queremos añadir una animación que solo puede hacerse en javascript y no en CSS tal vez deberíamos preguntarnos si de verdad es necesaria esa animación en la página.
5. No debemos utilizar tablas para maquetar, lo más óptimo es utilizar contenedores como “div”, aunque HTML 5 ya nos aporta contenedores semánticos mucho más óptimos para esta tarea “ARTICLE”, “SECTION”, etc.

Para lo que debemos utilizar las tablas es para el contenido que se encuentra tabulado.

6. También es muy importante la utilización de unidades relativas, como hemos podido comprobar en CSS podemos añadir unidades relativas como es “%” o “rem”, en lugar de utilizar unidades como “px”.

LOS TRES PILARES DE OPTIMIAZACIÓN DEL DISEÑO

RESPONSIVE

El diseño responsive no es solamente hacer que en una página web pueda verse en varias pantallas de tamaños diferentes, sino que va mucho más allá y para ello tenemos que ver tres técnicas responsive:

1. Técnicas aplicadas a todos los navegadores y sistemas operativos: como ya sabemos dependiendo del “browser” en el que abramos la página web, esta se verá de forma diferente porque no todos los navegadores aceptan todas las etiquetas HTML, por lo que, por mucho que trabajemos para desarrollar con uno de ellos en el resto se verá diferente.
Además de las diferencias que tienen entre sí los “browser”, también existen diferencias entre los diferentes sistemas operativos, a eso tenemos que sumarle que en la actualidad se consulta mucho internet a través de los teléfonos móviles y que cada teléfono móvil contiene también un sistema operativo diferente.
2. Técnica aplicada a que se pueda ver en todas las dimensiones y resoluciones de pantalla: este es el apartado más común cuando se habla de responsive, la utilización de media queries para que todos los ordenadores y móviles puedan ver los elementos dispuestos facilitando la lectura es lo que debemos utilizar.
3. También debemos pensar en todas las velocidades de conexión, hay personas que tienen el teléfono limitado con ancho de banda o planes de datos móviles que van ralentizados. Por esta serie de motivos tenemos que optimizar nuestro código, tanto HTML, como CSS, como javascript.
Esto es muy importante porque el usuario no espera más de 5 segundos para acceder a una página web.

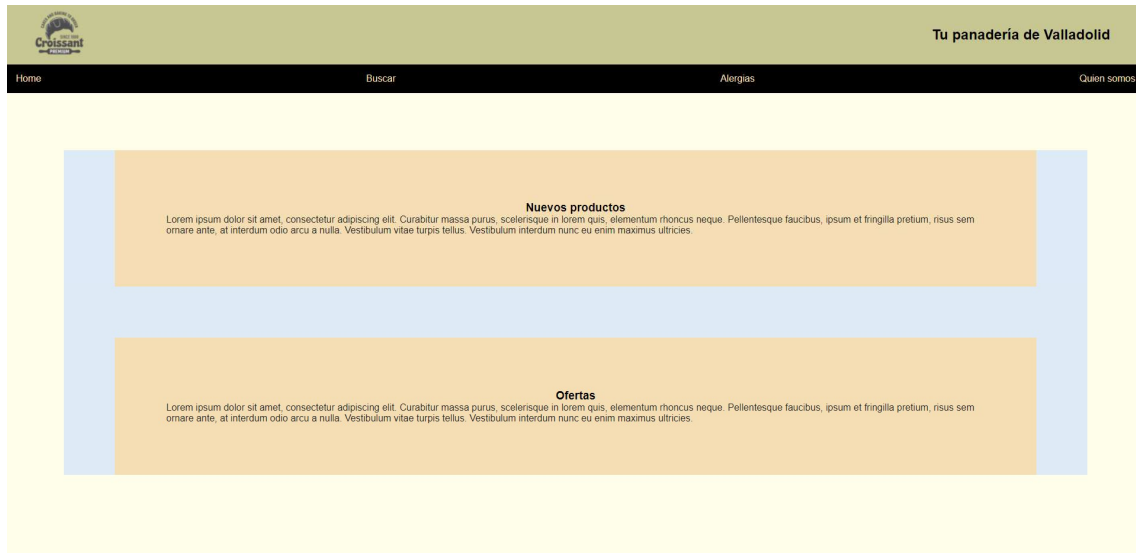
PASOS PARA DISEÑAR UN SITIO WEB RESPONSIVE

Lo que debemos de hacer antes de empezar a codificar un sitio web es tener un prototipo en papel, que tanto el cliente como el desarrollador hayan validado.

1. Lo primero que tenemos que hacer es crear un HTML con todo el contenido que deseemos que tenga la página, después le aplicamos una hoja de estilos CSS para aplicar el formato adecuado para presentar el contenido. El HTML debe ser sencillo pero con las funcionalidades necesarias, sobre todo en lo que bloques de contenido se refiere.

Lo primero que vamos a hacer es intentar encajar todo en una pantalla pequeña, porque si tenemos mucho contenido lo fundamental es que se pueda ver todo en la pantalla, en las pantallas más grandes no suele haber tanto problema.

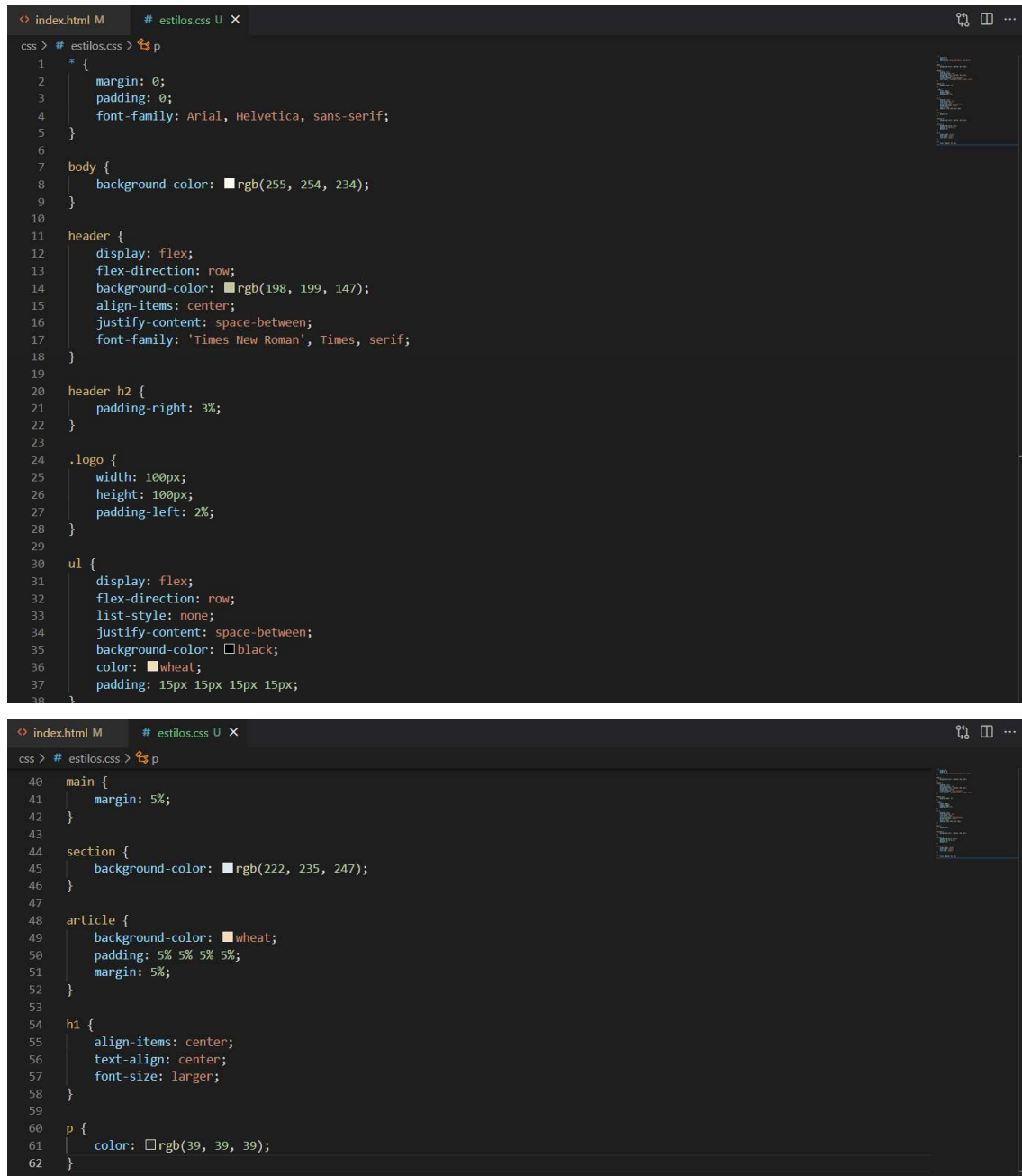
También es bastante interesante poner las imágenes que vayan en la cabecera o en el pie, en resumen, todo lo que se ha acordado anteriormente que la página debe contener.



```

index.html M X # estilos.css U
index.html > html > body > header > img.logo
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link rel="stylesheet" href="css/estilos.css">
8   <title>Responsive</title>
9 </head>
10 <body>
11   <header>
12     
13     <h2>Tu panadería de Valladolid</h2>
14   </header>
15   <nav>
16     <ul>
17       <li>Home</li>
18       <li>Buscar</li>
19       <li>Alergias</li>
20       <li>Quien somos</li>
21     </ul>
22   </nav>
23
24   <main>
25     <section>
26       <article>
27         <h1>Nuevos productos</h1>
28         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur massa purus, scelerisque in lorem quis, elementum rhoncus neque. Pellentesque faucibus, ipsum et fringilla pretium, risus sem
29       </article>
30       <article>
31         <h1>Ofertas</h1>
32         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur massa purus, scelerisque in lorem quis
33       </article>
34     </section>
35   </main>
36 </body>
37 </html>

```



```
css > # estilos.css > p
1  * {
2    margin: 0;
3    padding: 0;
4    font-family: Arial, Helvetica, sans-serif;
5  }
6
7  body {
8    background-color: ■rgb(255, 254, 234);
9  }
10
11 header {
12   display: flex;
13   flex-direction: row;
14   background-color: ■rgb(198, 199, 147);
15   align-items: center;
16   justify-content: space-between;
17   font-family: 'Times New Roman', Times, serif;
18 }
19
20 header h2 {
21   padding-right: 3%;
22 }
23
24 .logo {
25   width: 100px;
26   height: 100px;
27   padding-left: 2%;
28 }
29
30 ul {
31   display: flex;
32   flex-direction: row;
33   list-style: none;
34   justify-content: space-between;
35   background-color: □black;
36   color: ■wheat;
37   padding: 15px 15px 15px 15px;
38 }
39
40 main {
41   margin: 5%;
42 }
43
44 section {
45   background-color: ■rgb(222, 235, 247);
46 }
47
48 article {
49   background-color: ■wheat;
50   padding: 5% 5% 5% 5%;
51   margin: 5%;
52 }
53
54 h1 {
55   align-items: center;
56   text-align: center;
57   font-size: larger;
58 }
59
60 p {
61   color: □rgb(39, 39, 39);
62 }
```

2. Ahora tenemos que comenzar a crear el CSS es e incluirlo en el HTML, empezamos a crear y a introducir los estilos que debe tener la página, para que, de esta manera, se comience a ver como nosotros queramos.

Cada navegador soporta y entiende las reglas de las hojas de estilos de una manera diferente, si un navegador no soporta una regla de estilos no pasa nada porque simplemente la ignora.

- (a) **Recursos de diseño estéticos:** es el conjunto de recursos, como por ejemplo cajas, tipografías o textos que le añaden personalidad a un sitio web. Este conjunto de elementos puede hacer que nuestro sitio se vea mucho más bonito, pero por ejemplo, si el navegador da problemas y no lo

reconoce puede ser un fallo muy grave, porque a lo mejor desaparecen elementos que nosotros hemos añadido.

- (b) **Diseño de Layout:** es una plantilla en la que la información se presenta con una estructura definida, es decir, con columnas, cabecera, pie, etc. es muy importante que cuando un “browser” no es capaz de interpretar correctamente una web le dé prioridad a los elementos que tengan información importante para que el usuario pueda acceder a ellos, es decir, lo tenemos que decir a la página qué elementos son menos importantes y puede dejar de mostrar en caso de que haya algún problema.

DOS TÉCNICAS PARA RESOLVER PROBLEMAS

Progressive Enhancement: es un diseño que se centra en sistemas anticuados por lo que lleva menos soporte de CSS. Lo primero por lo que debemos preocuparnos es porque el sitio sea accesible, después tenemos que irle añadiendo complejidad aplicando CSS que soportan los navegadores actuales, y, como en los navegadores más antiguos no soportan CSS, simplemente no lo reconocerá.

Graceful Degradation: se recorre el camino en sentido inverso, es decir, desarrollamos el CSS atendiendo a las características más actuales de los navegadores nos centramos en que el sitio web se vea como deseamos, una vez estemos satisfechos con la web que hemos hecho pero vamos al sitio web el navegador es menos avanzados y se realizan las tareas necesarias para que el sitio web se vea de forma aceptable.

LA FILOSOFÍA MOBILE FIRST

La filosofía móvil first consiste en diseñar primero la interfaz que irá en el teléfono móvil, como ya hemos visto anteriormente es mucho más complicado diseñar una interfaz para un elemento que puede congrega muchos menos datos en la pantalla, por lo que, empezar a diseñar por aquí es una buena idea.

DISEÑO

Es la etapa en la que realizamos un prototipado para la interfaz del móvil, es mucho más sencillo realizarlo para un dispositivo que es mucho más pequeño.

Utilizar esta filosofía es cuestión de comodidad, de forma gradual a medida que vayamos trabajando con ello nos vamos acostumbrando a que las cosas cuando las pasamos a una pantalla más grande se vean de alguna forma más cutre. A medida que aumentamos el tamaño nos vamos dando cuenta de que entran muchas más cosas en la pantalla de un ordenador que en la de un teléfono móvil.



BREAKPOINT ORIENTADO AL DISEÑO RESPONSIVE

Los break points es como comúnmente se conoce a los saltos en el diseño responsive, es decir, cuando pasamos del tamaño de la pantalla de un teléfono móvil a la pantalla de un portátil y, a su vez, cuando pasamos de la pantalla de un portátil a una pantalla de un ordenador de sobremesa.

El diseño responsive destaca principalmente, por el hecho de que cuando la pantalla se va haciendo más pequeña el diseño comienza a cambiar, esto lo podemos ver en el modo desarrollador de cualquier navegador. Este tipo de saltos se producen gracias a las media queries.

Es muy importante no empeñarse en orientar estos puntos de ruptura a los diferentes tamaños de los distintos dispositivos, es imposible crear un punto de ruptura para cada dispositivo que existe, porque no todos los iPhones tienen el mismo tamaño, ni todos los ordenadores el mismo tamaño de pantalla.

¿Dónde colocar un breakpoint?

Para saber dónde colocar estos puntos lo que tenemos que hacer es colocar la pantalla como la pantalla más reducida que podamos, a medida que vamos estirando en la pantalla comenzamos a ver que el diseño se ve cada vez peor, es en este punto en el que ya prácticamente no se ve en el que debemos poner un breakpoint. Lo podemos comprobar con cualquier otra página web que sea adaptativa, hay un punto en el que la pantalla metió un breakpoint y se comienza a ver mucho mejor cambiando algunos elementos del diseño.

No hay que escatimar a la hora de poner breakpoints, podemos poner todos los que queramos y siempre se tiene que ver de forma correcta en la página que estamos desarrollando. Vamos a encontrarnos con que hay breakpoints muy grandes que cambiarán por completo el layout de la página, cuando esto sucede queda mucho más estético poner otros puntos intermedios para que no sea tan radical el cambio.

Twitter en sus librerías para maquetación utiliza las siguientes medidas para establecer los breakpoints, aunque lo más recomendable es realizar nosotros un estudio para ver cuáles son las más estandarizadas.

<576px -> Pantallas pequeñas

576-768px -> Móviles apaisados

768-992px -> Tablets

992-1200px -> Desktops

>1200px -> Pantallas grandes

Para colocar los breakpoint necesitamos utilizar las MediasQuerys, su sintaxis es "@media mediatype[condiciones]".

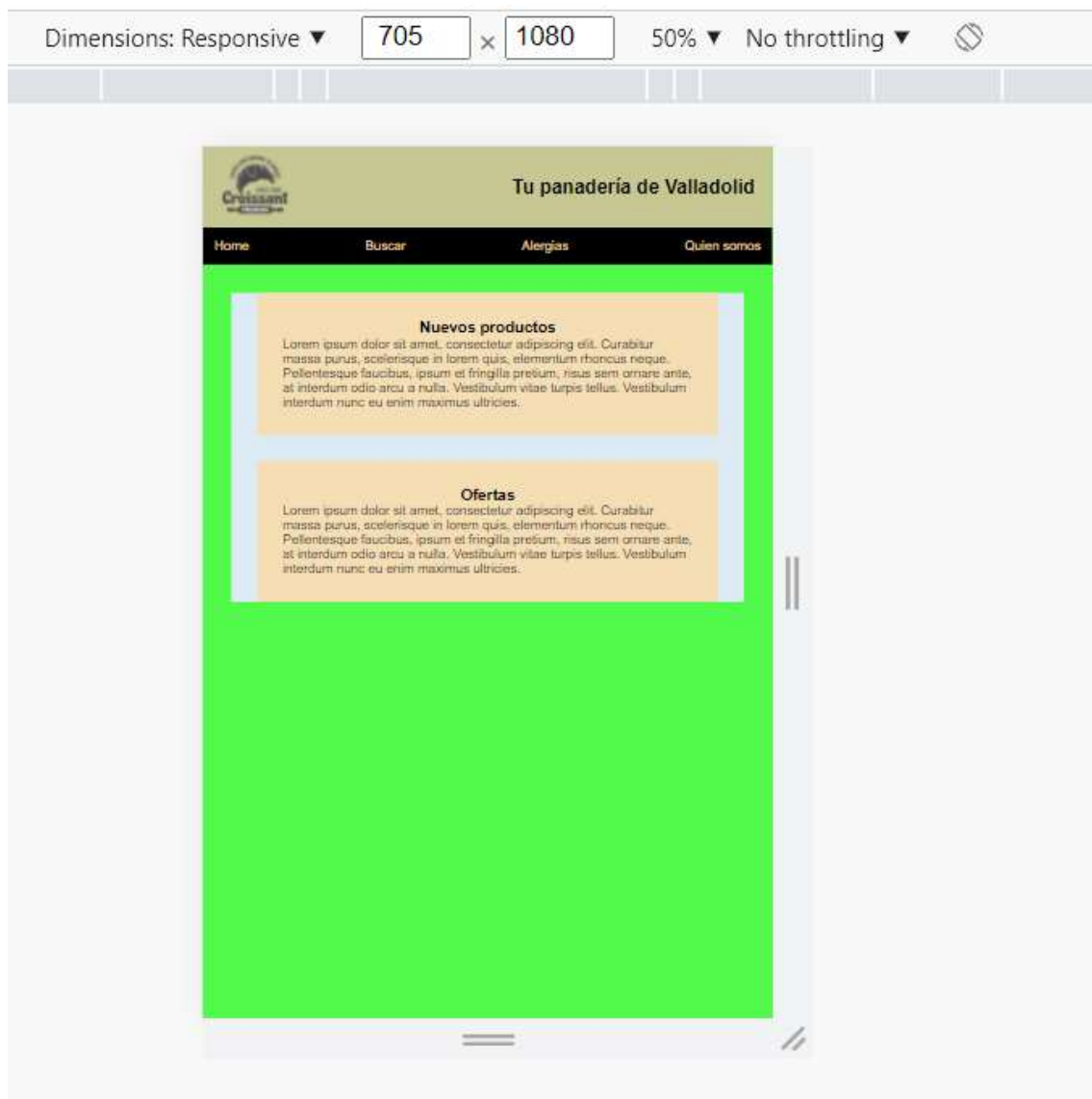
```
@media (min-width: 576px) and (max-width: 900px) {
```

Con esto establecemos entre que tamaño mínimo y que tamaño máximo tiene que aparecer el contenido que introduzcamos dentro.

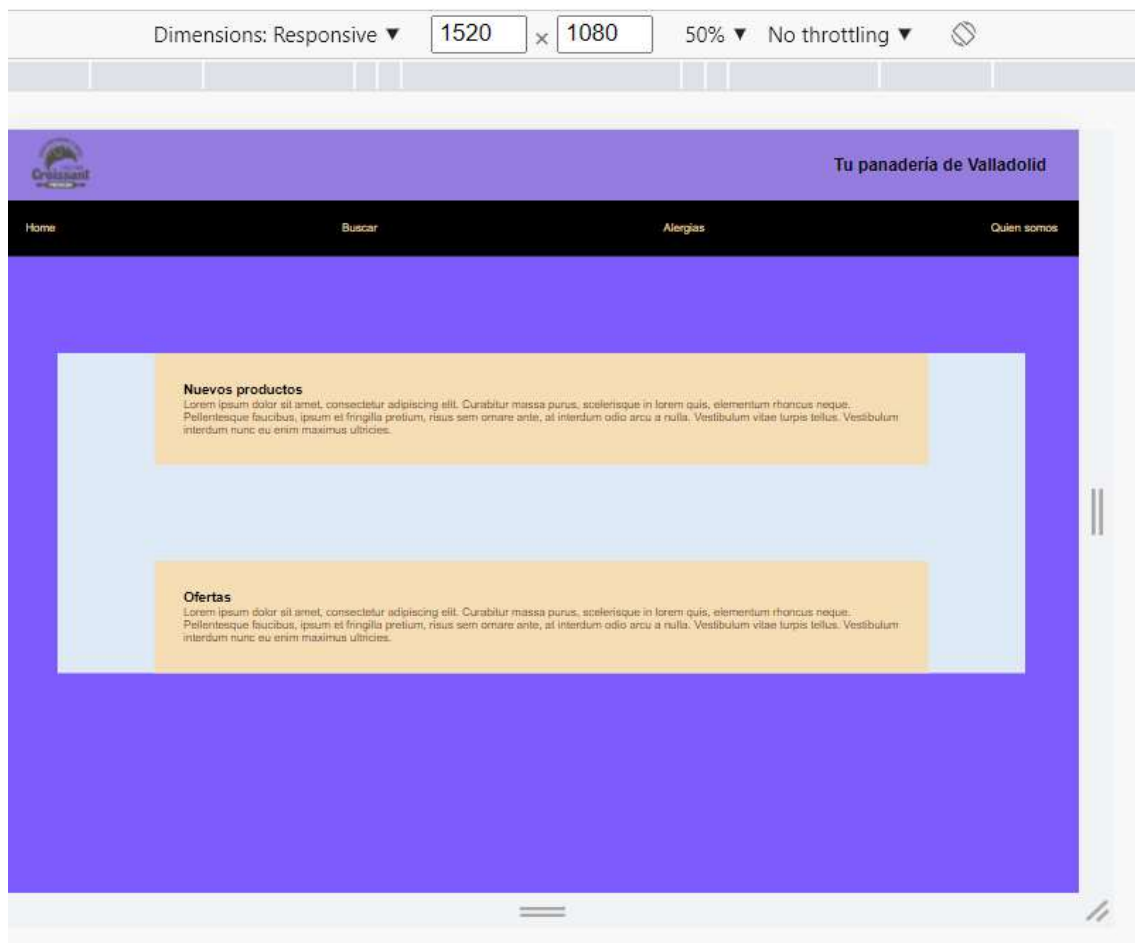
He introducido varios ejemplos, en los que cambia el color del fondo del body.

```
@media (min-width: 576px) and (max-width: 900px) {  
  body {  
    background-color: #546e7a;  
  }  
}
```

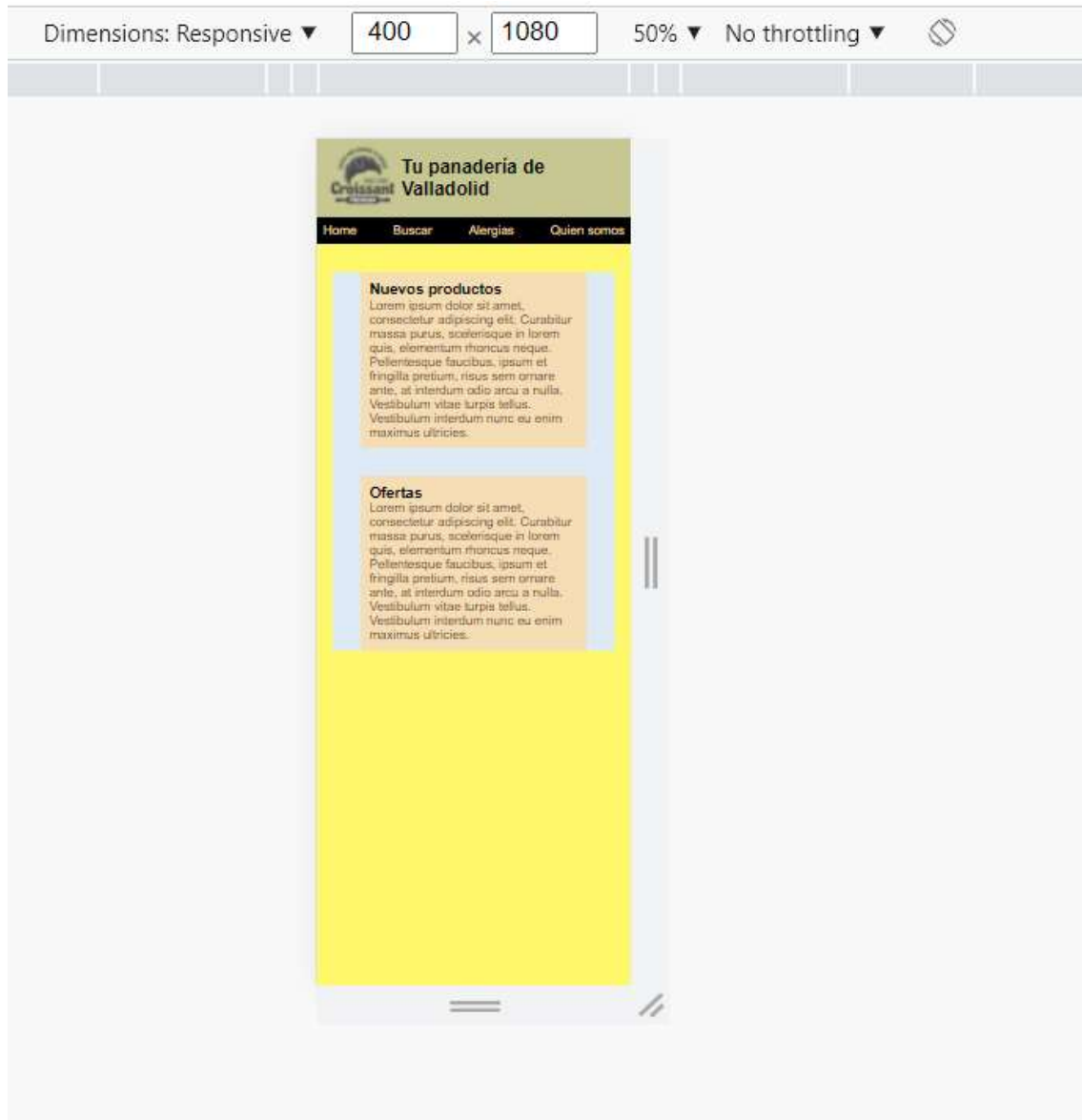
El resultado es este:



```
@media (min-width: 900px) and (max-width: 8000px) {  
  body {  
    background-color: #7b68ee;   
  }  
}
```



```
@media (min-width: 100px) and (max-width: 576px) {  
  body {  
    background-color: #ffef99;  
  }  
}
```



UNIADES DE MEDIDA EN CSS

Existen dos tipos de medidas en CSS:

UNIDADES FIJAS: son las que especifican una medida en términos absolutos, no tienen en cuenta el contexto en el que se está aplicando, un ejemplo son las medidas en pixeles

UNIDADES RELATIVAS: son las que tienen en cuenta el contexto en el que se encuentran, se consideran relativas al contenedor en el que se han definido, un ejemplo de unidades relativas es el "%". Las medidas son relativas al contenedor en la que se encuentran, es decir, en un contenedor de tipo "div" definimos el tamaño de la fuente a "25px", si le aplicamos una medida relativa del 200%, se hará en base a la unidad definida anteriormente.

Otra de las unidades relativas más utilizadas es “em”, esta unidad funciona también de forma relativa al tamaño de los contenedores, es decir, que si a un elemento que se encuentra dentro de un contenedor le aplicamos la medida relativa “2em”, habrá que multiplicar por dos el tamaño del contenedor.

Las unidades más recomendadas a la hora de realizar un diseño responsive, son:

- EM: Recordamos que si a un contenedor le asignamos 20px, 1em corresponderá a la medida anterior.
- REM: también es relativo al tamaño asignado anteriormente, pero esta vez nos referimos al tamaño asignado a la fuente del elemento HTML. El problema que tiene esta medida es que en navegadores más antiguos no se reconoce.
- VW y VH: son de las medidas más recientes y su significado corresponde a “viewport-width” y “viewport height”. 1vh es la centésima parte de la altura del viewport. Nos sirve para cuando hay elementos que necesitamos que ocupen partes enteras de la pantalla.

ESTILOS CSS BÁSICOS PARA RESPONSIVE WEB DESIGN

Box-sizing: este atributo precedente CSS3 y lo que nos permite es que el intérprete, es decir, el navegador, interprete las medidas del viewport de forma diferente a cómo lo haría de forma normal ese navegador. Los navegadores calculan la pantalla como si de las dimensiones de una caja se tratase, el único problema que tiene este método es que algunos navegadores tienen en cuenta el padre y otros no, pasa lo mismo con el borde.

Flexbox: otra de las opciones que tenemos para maquetar de una forma mucho más versátil, rápida y evitando muchos de los problemas a la hora de posicionar elementos es utilizar Flexbox. El único problema que tiene es que hay navegadores más antiguos que todavía no lo soportan, pero sabemos que nuestra aplicación web no va a verse en ninguno de estos navegadores, lo óptimo es utilizar Flexbox.

Atributos min-width, max-width/min-height, max-height: con este tipo de atributos definimos las alturas máximas y mínimas, a la vez que la anchura máxima y mínima. Aquí se suele utilizar medidas como los píxeles y podemos utilizarlo, por ejemplo, para imágenes.

LA ETIQUETA VIEWPORT

Primero tenemos que entender que es un viewport, un viewport es absolutamente todos los elementos que aparecen a la vez en el espacio disponible que nos ofrecen una pantalla.

Dentro del CSS3 existe una etiqueta viewport desde la que podemos cambiar el navegador desde el que estamos accediendo al sitio web.

También podemos modificar esta etiqueta para conseguir que el navegador se comporte como nosotros queremos, tenemos una serie de etiquetas que nos permiten hacerlo:

1. **Width:** anchura virtual (emulada) de la pantalla o anchura del viewport.
2. **Height:** altura virtual de la pantalla o anchura del viewport.
3. **Initial-scale:** la escala inicial del documento.
4. **Minimum-scale:** la escala mínima que se puede poner en el documento.
5. **Maximum-scale:** la escala máxima configurable en el documento.
6. **User-scalable:** si se permite o no al usuario hacer zoom.

EL ELEMENTO PICTURE

Este elemento es simplemente una etiqueta, no representa contenido, se utiliza para insertar varias imágenes en un mismo documento.

Dentro de esta etiqueta añadimos las imágenes con la etiqueta SOURCE, estas etiquetas deben contener un atributo “media” desde el cual se debe indicar una media query CSS.

EL ELEMENTO SOURCE

Es una etiqueta que nos permite indicar cada una de las imágenes que vamos a añadir, lo que nos permite es indicar archivos gráficos y el navegador se encargará de irlos leyendo uno a uno, siempre desde el primero hasta el último.

srcset: es el atributo que contiene el archivo que debe visualizarse, podemos indicar alternativas de imágenes separadas por comas y podemos indicar versiones de la misma para distintas densidades de píxeles.

Media: este atributo es capaz de soportar cualquier media query que podemos seleccionar con el atributo “media”. Es interesante que alguno de estos SOURCE se adapta a las reglas del medio query.

Sizes: este atributo ayuda a realizar la selección de la imagen apropiada de manera distinta, aunque utilizando este atributo delegamos más al navegador la elección de la imagen lo que debe ahorrarnos problemas.

Con los tamaños podemos indicar al navegador la anchura de la imagen y dejar que este escoja la imagen en función de los archivos proporcionados.

Type: este atributo indica el tipo de imagen, de modo que el navegador pueda seleccionarlo si reconoce ese formato de archivo.

```
<picture>
  <source srcset="img/florespqn.png" media="(min-width: 100px)">
  <source srcset="img/floresgrn.png" media="(min-width: 1000px)">
  
</picture>
```

DETECTAR MEDIA QUERIES CON JAVASCRIPT

No es necesario que utilicemos javascript para detectar las media queries, pero en caso de que queramos mostrar, por ejemplo, un banner solo en la versión de escritorio pero con la versión para móvil no aparezca, necesitamos la ayuda de javascript.

EL MÉTODO WINDOW MATCHMEDIA

Este es el método de comprobación que utilizaremos para así un media query se está cumpliendo.

Devuelve un objeto “mediaquery list” y nos informa de que se ha comprobado el valor.

Para saber si se cumple debemos hacer un “mediaqueryList.matches”.