CSS - Diseño responsivo o reactivo

Tabla de contenido

Estilos alternativos

Además de poder usar en una página varios archivos de estilo que se complementen y diferentes estilos según el medio usado, como veremos en la siguiente sección, tenemos la posibilidad de definir estilos alternativos que el usuario podrá elegir para cambiar el aspecto visual de nuestras páginas. El problema es que, de los navegadores principales, solo Firefox deja cambiar de estilos mediante el menú Ver > Estilo de página; en los demás debemos usar JavaScript para ello.

Estilos persistentes

Siempre están activados. Se usan para los estilos comunes y son los que hemos venido usando hasta ahora. No llevan atributo **title** como veremos que si llevan los demás casos.

Sintaxis:

```
<!-- Hoy día ya no se suele poner type="text/css" -->
<link rel="stylesheet" type="text/css" href="nombre.css" />
```

Estilos preferidos

Están activados por defecto. Se definen como los anteriores pero indicando un atributo title para poder elegirlos.

Sintaxis:

```
<link rel="stylesheet" type="text/css" href="nombre.css" title="titulo" />
```

Podemos crear varios estilos con un mismo título para formar un grupo y activarlos / desactivarlos a la vez.

Estilos alternativos

Están desactivados por defecto.

Sintaxis:

```
<link rel="alternate stylesheet" type="text/css" href="nombre.css" title="titulo" />
```

Algunos navegadores también admiten el valor alternative. También se les pueda dar el mismo título a varios para agruparlos y poder escogerlos en grupo.

Ejemplo completo

index.htm

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>estilos</title>
    <link rel="stylesheet" href="comunes.css">
    <link rel="stylesheet" href="preferidos.css" title="preferidos">
    <link rel="stylesheet" href="oscuros.css" title="oscuro">
    <link rel="stylesheet" href="claros.css" title="claro">
</head>
<body>
    <div>Lorem ipsum dolor sit amet.</div>
</body>
</html>
```

comunes.css

```
div {
    font-weight: bold;
    font-size: 2rem;
}
```

preferidos.cs

```
div {
    background-color: rgb(128, 236, 128);
    color: blakc;
}
```

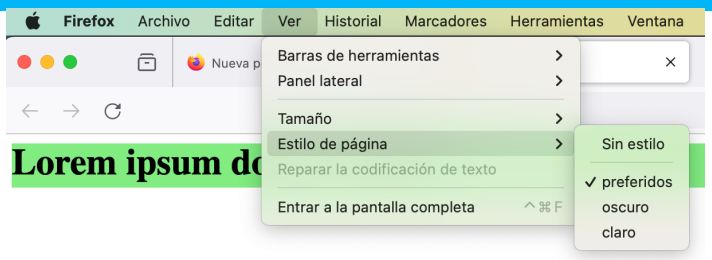
oscuros.css

```
div {
    background-color: black;
    color: white;
}
```

claros.css

```
div {
    background-color: lightgray;
    color: black;
}
```

- el texto va en negrita y 2rem en todos los estilos salvo que se definan otros en los estilos alternativos y tengan mayor especificad
- por defecto el texto irá en negro y en color verde de fondo
- si elijo el estilo oscuro, el texto irá en blanco y el fondo en negro
- si elijo el estilo claro, el texto irá en negro y el fondo en gris claro



Estilos para diferentes medios

CSS nos permite definir estilos diferentes según el medio donde se vaya a mostrar una página. Para ello usaremos el atributo **media** dentro de la etiqueta link.

Algunos de los posibles valores para este atributo son (se puede poner más de uno separado por comas):

- all: para todos. Es el valor por defecto
- aural: lectores y dispositivos de audio
- braille: si se muestra en dispositivos Braille
- embossed: para imprimir en impresoras Braille
- handled: si se muestra en una pda o móvil con navegador. Hoy día ya no se usa pues los móviles han mejorado tanto que no necesitan un estilo de este tipo
- print: si se imprime o se muestra la vista previa
- projection: si se muestra en un proyector
- screen: si se muestra en un monitor. Es la opción por defecto si no se indica ninguna
- tv: si se muestra en una televisión
- speech: reemplaza hoy dúa a aural.

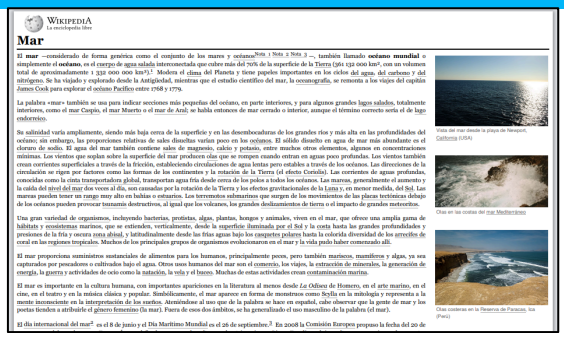
Los más usados, y soportados por la mayoría de los navegadores, son all, screen, speech y print; este último nos permite modificar el diseño de la página a la hora de imprimirla (normalmente eliminando contenido como publicidad, menús, colores de fondo, …, usando en muchos casos la propiedad display a none. También se suelen indicar medidas en unidades como puntos, que se adecúan bien para imprimir). Aquellos estilos que no queremos que se apliquen a la hora de imprimir debemos definirlos con el valor screen (los navegadores, por defecto, no suelen imprimir colores o imágenes de fondo). Los estilos a los que no indiquemos atributo media se usarán en todos los casos.

Es el navegador o dispositivo usado el que automáticamente elige el estilo adecuado en cada caso.

Ejemplo:

```
<link rel="stylesheet" href="imprimir.css"  media=”print" />
```

En el siguiente ejemplo podemos ver una página web de la Wikipedia tal como se muestra en el navegador y luego en la vista previa de impresión:



Con import también podemos indicar el medio deseado después del paréntesis.

Ejemplo:

```
@import url(imprimir.css) print;
```

También podemos indicar el medio a usar dentro de una etiqueta style, aunque siempre será mejor en un archivo CSS por separado.

Ejemplo:

```
<style type="text/css">
     @media print {
       body { font-size:12pt;}
     }
     @media screen {
       body {font-size:1em; }
     }
</style>
```

Diseño responsivo

Con el auge de los teléfonos móviles inteligentes, los diseñadores web se encontraron con la necesidad de diseñar páginas web para los mismos. En un principio existían muchas limitaciones de tamaño de pantalla, CPU, memoria RAM y velocidad de conexión en los mismos, lo que obligaba a diseñar páginas más sencillas para adaptarnos a estas limitaciones, además de que los navegadores eran bastante más limitados que sus versiones de ordenador.

Hoy día, al menos en los países más avanzados, esas limitaciones han casi desaparecido, teniendo dispositivos móviles bastante potentes y con pantallas cada vez más grandes. Es más, la mayoría de los móviles actuales tienen más densidad de píxeles y resolución que muchas de las pantallas de los ordenadores que usa la gente. Por ejemplo, un Xiaomi Redmi Note 12, de unos 100 euros, tiene una resolución de 2400x1080 píxeles mientras que muchos PCs, sobre todo portátiles, siguen en 1920x1080 o menos.

Aun así debemos tener en cuenta las siguientes consideraciones:

- los móviles son más altos que anchos, con lo que el consumo habitual de contenido es en vertical, no en horizontal como en los PCs. Por ello, desplazarse para ver el contenido verticalmente es algo que se considera normal, horizontalmente no.
- pese a tener grandes resoluciones, es en un espacio más pequeño, con lo que todo se verá más pequeño con respecto a un PC

- la duración de la batería de los móviles sigue siendo algo a tener en cuenta, sobre todo si un usuario va a visitar durante mucho tiempo o veces nuestra web.
- también hay que tener en cuenta que en los dispositivos móviles no se suele usar ratón, con lo que determinados efectos o funcionamientos diseñados para un ratón no funcionarán.

Por ello hay diferentes maneras de tener esto en cuenta:

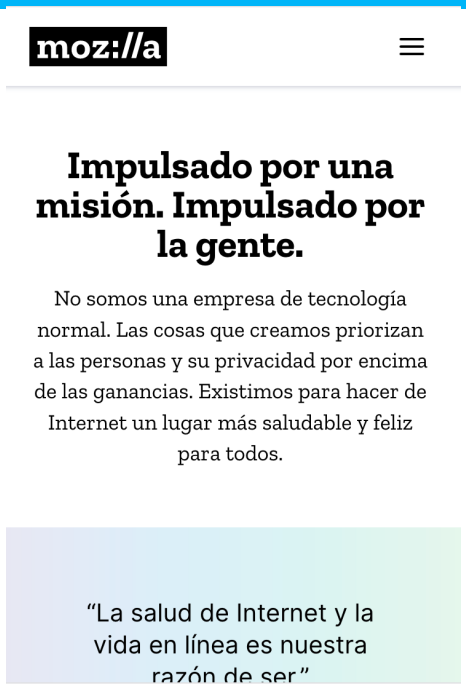
- crear la página solo para móvil o para PC. No será lo habitual pero puede que haya casos en los que tenga sentido: el público objetivo de la página, la complejidad de esta en cuanto a elementos que se necesitan ver al mismo tiempo, ...
- crear páginas diferentes para diferentes dispositivos, normalmente móviles (mobile) o escritorio / PC (destokp). Esto puede facilitar la creación de las páginas, sobre todo en casos complejos, pero puede hacer más costoso el mantenimiento
- mobile first: crear las páginas teniendo en cuenta primero los dispositivos móviles y luego adaptarla a PCs. Como hoy día cada vez más gente usa el móvil, esta suele ser una opción muy usada. Se centra en el contenido primero
- desktop first: lo contrario, creamos primero la página para PCs y luego la adaptamos para móviles. En ocasiones se elimina contenido en las versiones de móviles; por ejemplo, con display: none. Hoy día no se considera adecuada en la mayoría de los casos pues considera a los móviles ciudadanos de segundo, cuando la realidad es otra

Para los dos últimos casos hoy día se usa mucho el diseño responsivo (RWD) donde usaremos diseños fluidos que se adapten a todo tipo de dispositivos, usando disposiciones flexibles, imágenes incluidas, unidades relativas, uso de media querys CSS, ... hasta la tipografía puede ser responsiva. La aparición de Flexbox y Grid han simplificado mucho este proceso, pues anteriormente teníamos que basarlo casi todo en el uso de float.

Ventajas e inconvenientes del diseño responsivo:

- se ofrece una experiencia mejor al usuario al tener la web adaptada a diferentes dispositivos. Eso sí, siempre que hagamos bien las cosas, que hay webs que, por simplificar, hacen la experiencia más molesta
- se evita tener que crear páginas para diferentes lo que complica el mantenimiento
- lo anterior también implica el ahorro de costes de mantenimiento, aunque al principio sean mayores por la mayor dificultad que puede ser preparar la web responsiva en casos de webs complejas. Por ejemplo, los menús de navegación son más complejos de implementar en móviles (típicos menús hamburguesa)

La web de Mozilla desde un PC



La misma web accediendo desde un móvil

Otras definiciones:

- degradación agraciada (Grateful degradation): consiste en diseñar la página para las últimas tecnologías y luego ir simplificándola para que siga funcionando en navegadores o dispositivos más limitados
- mejora progresiva (Progressive enhancement): es lo contrario de lo anterior, partimos de la versión más simple y vamos añadiendo más funcionalidad
- pixel perfect: consiste en crear una página que sea exactamente igual o muy similar al prototipo

Resolución del dispositivo vs resolución de pantalla o resolución CSS

La resolución de un dispositivo es el número real de píxeles que tiene; algo simple y sencillo; son los que hay y no se pueden cambiar.

Hoy día muchos dispositivos tienen mucha resolución con lo que pueden encontrarse con el problema de que, por ejemplo, una imagen de un determinado número de píxeles podría quedar bien en un dispositivo con una resolución y mucho más pequeña en otros con más resolución.

Por ejemplo, en Apple, cuando sacaron el iPhone 4, este tenía una resolución del dispositivo de 640 x 960 frente a los 320 x 480 del iPhone 3GS ¿Qué pasó entonces con todas las aplicaciones existentes? ¿Pasaron a ocupar la mitad de la pantalla y viéndose todo más pequeño (pues los dpi eran mayores al no cambiar el tamaño físico de pantalla)? Lo que hizo Apple es que cada píxel se escalaba el doble en este nuevo iPhone, lo que se conoce como factor de escala o ratio de píxeles, con lo que las aplicaciones ocupaban lo mismo en pantalla. Hoy día, con dispositivos con mayores dpis, el factor ha pasado a 3 o 4 en los dispositivos más modernos.

Una solución simple pero que trae un inconveniente cuando hablamos de imágenes o videos. Como se tiene que mostrar cada píxel en más píxeles, la imagen no quedará tan definida. La solución pasó a tener que incluir imágenes en más de una resolución para que se viese nítida en ambos dispositivos.

Por ejemplo, para una imagen de 200x200 teníamos que incluir otra de 400x400; ambas ocuparían lo mismo en pantalla pero la segunda se vería más nítida en dispositivos con más píxeles reales.

Hoy día, en ordenadores con mayor densidad de píxeles también pasa esto, aunque no se suele pasar de 2 de ratio. Por ejemplo un MacBook Air M1 tiene una resolución de dispositivo de 2880 x 1800 pero 1440 x 900 de pantalla, es decir, un ratio de 2.

En estas páginas web podemos encontrar el ancho y alto en píxeles reales, el ratio de aspecto, la densidad, ... de muchos dispositivos móviles y portátiles:

https://viewportsizer.com/devices/
https://screensiz.es/

Y en esta la de nuestro ordenador o dispositivo:

https://www.whatismyscreenresolution.org/

Meta viewport

Todas nuestras páginas deberían llevar la siguiente etiqueta en el head:

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

Esta indica al navegador que establezca el ancho del viewport al ancho del dispositivo, debido a que los navegadores de los móviles solían mentir con ese tamaño para poder mostrar la página a tamaño completo, reduciendo el zoom, pues las webs no estaban preparadas para ellos. Esto hacía que el contenido de las páginas normalmente se viesen muy pequeño y el usuario tenía que andar haciendo zoom y desplazándose para ver el contenido.

Consultas de dispositivos / Media Querys

En CSS2 podíamos indicar que un archivo de estilos se aplicará a la pantalla, a dispositivos móviles, a impresión, ... Ahora en CSS3 tenemos muchas más posibilidades con las llamadas *media queries*. Estas nos permiten aplicar estilo basándonos en la orientación, ancho y alto del dispositivo, resolución, ... lo cual es muy importante para crear un diseño responsivo.

Podemos usarlas dentro de los estilos, en etiquetas *link y sentencias import*. Vamos a verlas con ejemplos pues tenemos muchas posibilidades y combinaciones posibles.

```
/* Dispositivos con una ventana con un ancho máximo de 480px */
@media only screen and (max-width: 480px){
    /* propiedades */
}

/* Dispositivos con un ancho mínimo de ventana de 640px */
@media all and (min-width: 640px) {
    /* Propiedades */
}

/* Dispositivos donde el usuario prefiere el modo oscuro (la otra opción es light) o con un máximo de
ventana de 900px */
@media all and (prefers-color-scheme: dark), (max-width: 900px){
    /* Propiedades */
}

/* Dispositivos con un ancho de ventana entre 600px y 900px, ambos incluidos
```

```
@media all and (min-width: 600px) and (max-width: 900px){
    /* Propiedades */
}

/ * Igual que el anterior pero una forma más simple añadida posteriormente */
@media all and (600px <= width <= 900px) {
    /* Propiedades */
}

<!-- Todos los dispositivos con orientación apaisada -->
<link rel="stylesheet" media="all and (orientation: landscape)" href="estilo.css">

<!-- Todos los dispositivos con orientación vertical -->
<link rel="stylesheet" media="all and (orientation: portrait)" href="estilo.css">

<!-- Dispositivos de pantalla con un ancho de ventana de 480px y una altura de 900px */
<link rel="stylesheet" href="estilo.css" media="only screen and (width: 480px) and (height: 900px)">

<!-- Impresoras en color -->
<link rel="stylesheet" media="print and (color)" href="estilos.css">

/* Pantallas e impresorsas */
@media screen, print {
    /* Propiedades */

}
/* Dispositivos de pantalla con un ancho máximo de la ventana de 480px y una resolución de 163 dpis */
@import url("estilos.css") screen and (max-width: 480px) and (resolution: 163dpi);

/* Impresoras que no sean de color */
@import url("estilos.css") print and ( not( color ));
```

Notas:

- también podemos encontrarnos con las opciones min-device-width, max-device-width, device-height, ... que se refieren al tamaño completo de la pantalla, no de la ventana. Ya no se usan hoy día
- existen muchas más opciones, pero las vistas en los ejemplos son las más usadas
- en todos estos casos hemos usado píxeles pero pueden ser otras unidades de medida
- podemos anidar media querys:

  Ejemplo:

```
/* Cuando el tamaño sea de un ancho menor o igual a 900px, el texto se pondrá a 2rem y de color rojo. En caso de mayor tamaño será negro y de 1ren (suponiendo que no hay nada cambiado) */
@media all and ( max-width: 900px) {
    div {
        font-size: 2rem;
        color: green;
    }
    /* Cuando esté entre 700 y 900 píxeles irá en color rojo; por debajo de 700 irá en verde */
    @media (min-width: 700px ) {
        div {
            color: red;
        }
    }
}
```

Unidades de viewport

Aparecieron con CSS 3 para darnos control con respecto al tamaño de la ventana, coincidiendo en muchos móviles con el de la pantalla.

Son las siguientes:

- vw: un porcentaje del ancho del viewport. Por ejemplo 10vw, es un 10% del ancho
- vh: un porcentaje del alto del viewport
- vmin: un porcentaje del mínimo del viewport, ya sea el ancho o alto. Por ejemplo 10vmin, en un móvil suele ser un 10% del ancho con la pantalla apaisada / horizontal; un 10% del alto con la pantalla en vertical, aunque hay algunos móviles que permiten tener más de una app a la vez en pantalla, con lo que no sería siempre así
- vmax: un porcentaje del máximo del viewport, ya sea el ancho o alto

La principal diferencia con el uso de porcentajes es que estos últimos se refieren a su contenedor, mientras que vw, vh, ... se refieren a la ventana.

Podemos usarlos en tamaños, fuentes, márgenes, bordes, ...

Ejemplo:

```
/* Todo el contenedor será negro de ancho, pues será un 50% por cada lado */
border: 50vw solid black;
```

Encabezado y pie fijo con el contenido desplazándose

Una manera de conseguir esto es:

- height: 100vh en el body para que coja todo el alto de la ventana
- display: grid en el body para colocar el encabezado, contenido y pie. También podría usarse flex
- grid-template-rows: auto 1fr auto; en el body para que ocupe el contenido todo el espacio sobrante
- overflow: scroll en los contenidos para que, si ocupan más espacio del que sorba entre el encabezado y pie, podamos hacer scroll por el mismo y que no quede contenido oculto

El pie se pondrá siempre en la parte inferior aunque el contenido no ocupe todo el ancho.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>responsivo</title>
    <style>
        body {
            height: 100vh;
            font-size: 2rem;
            display: grid;
            grid-template-columns: 1fr 1fr;
            grid-template-rows: auto 1fr auto;
            gap:20px
        }
        div {
            /* Para que el contenido se desplaze haciendo scroll */
            overflow: scroll;
        }
        header, footer {
            grid-column: 1 /3;
```

```
            background-color: black;
            color: white;
            padding: 10px;
        }

    </style>
</head>
<body>
    <header>Encabezado</header>
    <div>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab illum at, minus ex, accusantium eligendi vero odio, provident quasi sequi tenetur error. Atque veritatis nulla enim, dignissimos soluta obcaecati rem debitis saepe quos itaque fuga, fugit consectetur nisi neque earum eveniet. Eos atque nihil officia, voluptatibus facere alias, nostrum id, nemo laboriosam ducimus ipsum exercitationem ullam ea quisquam! Atque ex hic ad esse sed iure odit quidem commodi facere fugit nam quisquam doloribus fugiat ipsa assumenda, asperiores sapiente a officiis inventore blanditiis illum recusandae at? Dolorem ea accusantium inventore, eius ipsam quam facere eligendi magnam doloremque molestiae vel quod repellat.</div>
    <div>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ab illum at, minus ex, accusantium eligendi vero odio, provident quasi sequi tenetur error. Atque veritatis nulla enim, dignissimos soluta obcaecati rem debitis saepe quos itaque fuga, fugit consectetur nisi neque earum eveniet. Eos atque nihil officia, voluptatibus facere alias, nostrum id, nemo laboriosam ducimus ipsum exercitationem ullam ea quisquam! Atque ex hic ad esse sed iure odit quidem commodi facere fugit nam quisquam doloribus fugiat ipsa assumenda, asperiores sapiente a officiis inventore blanditiis illum recusandae at? Dolorem ea accusantium inventore, eius ipsam quam facere eligendi magnam doloremque molestiae vel quod repellat.</div>
    <footer>Pie de página</footer>
</body>
</html>
```

Tamaños mínimos y máximos: min, max y clamp

Las propiedades min-width, min-height, max-width y max-height que ya conocemos, nos serán también muy útiles para hacer nuestros elementos responsivos.

La función min escoge el menor valor de una serie de expresiones separadas por comas, mientras que la función max coge el mayor. En los valores se pueden anidar otras funciones max, min y calc.

Ejemplo:

```
width: min(40%, 400px);
font-size: max(4vw, 2em, 2rem);
```

La función clamp admite tres valores: el mínimo, el preferido y el máximo, donde se usará el valor preferido siempre que no sea menor que el mínimo ni mayor que el máximo.

Ejemplo:

```
font-size: clamp(1rem, 2.5vw, 3rem);
```

Fuentes responsivas

Uno de los usos de las unidades de viewport es para indicar el tamaño de una fuente y que esta se adapte al tamaño de la ventana.

Ejemplo:

```
div {
    font-size: 5vw;
}
```

Los problemas de este método son dos:

- el tamaño puede crecer o decrecer demasiado
- el usuario no puede cambiar el zoom del texto correctamente

Una alternativa es la siguiente:

```
div {
    /* AUmentamos o disminuimos el tamaño
    font-size: calc(1rem + 1vw);
    line-height: calc(1.1rem + 0.5vw);
}
```

También podemos hacer uso de las media queries para cambiar el tamaño según alguna condición.

Ejemplo:

```
body {
  font-size: 100%;
}

h1 {
  font-size: 2rem;
}

@media (min-width: 1000px) {
  h1 {
    font-size: 4rem;
  }
}
```

Imágenes responsivas

La "responsibidad" de las imágenes podemos verla de dos formas:

- que adapten su tamaño al de la ventana, agrandándose o empequeñeciéndose hasta ciertos límites
- que se sirva una imagen u otra dependiendo de los dpis o tamaño a ocupar.

Para todo esto lo ideal sería usar un formato de imagen como SVG, que no pierde calidad al cambiar de tamaño, pero muchas veces no será posible.

Una opción muy habitual es establecer el width o max-width a un 100% para que la imagen se adapte siempre a su contenedor, recordando que la altura se ajusta también automáticamente.

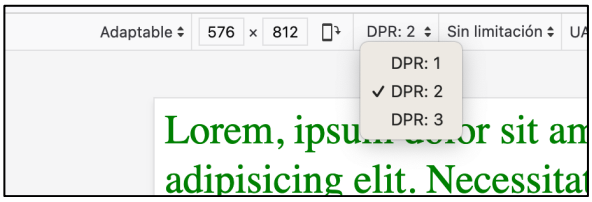
Otra de las opciones es usar el atributo **srcset** de la etiqueta img que ya hemos visto en su día. Este nos permite indicar la imagen a usar dependiendo de los dpis (puntos por pulgada) de la pantalla.

Ejemplo:

```
<img
  src="casa.jpg"
  srcset="
    casa2x.jpg 2x,
    casa3x.jpg 3x,
  "
  alt="casa."
>
```

En este caso, en dispositivos se muestra la imagen casa.jpg, casa2x.jpg o casa3x.jpg, dependiendo de los dps; a más número mayor densidad. Hoy día casi todos los dispositivos son 2x al menos.

En las herramientas de desarrollo de Firefox o Chrome, cuando elegimos ver la página con diferentes dispositivos, tenemos la elección de elegir o ver los dpis del dispositivo.



Otra opción es indicar anchos en lugar de densidades, con lo que el navegador escogerá la más adecuada dependiendo del tamaño.

Con esto podemos evitar que se cargue una imagen demasiado grande si la ventana tiene un tamaño menor.

También puede ser útil para mostrar otras imágenes. Por ejemplo, mostrar una imagen recortada con la parte importante para que se vea más.

Ejemplo:

```
<img src="gente.jpg"
    srcset="
        gente_zoom.jpg 500w,
        gente.jpg 100w,
        "
    max-width="100%"
    alt="paisaje">
```

Imagen sin recortar

Imagen recortada