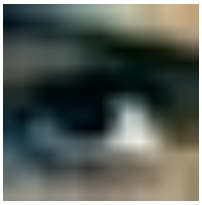
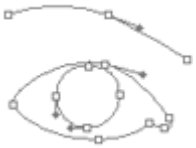
LAS IMÁGENES EN LA WEB (Una imagen vale más que 1,5 millones de palabras)

La elección y el diseño de las imágenes es una de las decisiones más importantes en el diseño web, deben ser de calidad, deben pesar poco, el conjunto debe ser homogéneo, la información que aportan debe ser accesible, deben ser responsivas, agradables, funcionales, entendibles, deben estar en consonancia con los contenidos de la web, no debemos vulnerar los derechos de autor. etc. Etc. Etc.

Pero..¿de donde sacamos las imágenes?..Pues tenemos varias posibilidades, podemos fabricarlas nosotros, podemos copiarlas, podemos copiarlas y adaptarlas, podemos utilizar la iconografía de un framework y también podemos comprarlas.

Una de las principales decisiones a la hora de incluir imágenes en una web es elegir el formato correcto para cada tipo de imagen de manera que se logre una correcta relación entre la calidad visual de la misma y su peso en bytes. Básicamente, existen distintos tipos de imágenes: las que son mapas de bits, las vectoriales y las imágenes animadas. Estas últimas las trataremos más adelante en la sección dedicada a las animaciones. Esta sección se centra en los otros dos grupos de imágenes (vectoriales y mapas de bits).





Pixeles

Los pixeles son la unidad mínima de las imágenes de **mapas de bits**, que también son llamadas imágenes **raster** o **bitmaps**.

Un mapa de bits es una matriz cartesiana (bidimensional) de pixeles, con coordenadas verticales y horizontales que determinan la posición de un pixel en la imagen.

Vectores

Los vectores son la **descripción geométrica** (matemática) de una imagen.
Por ejemplo, para describir todos los puntos del perímetro de un círculo sólo es necesaria su fórmula ($x^2 + y^2 = R$). Modificando la variable R, se obtienen círculos de todos los radios posibles.

Un ejemplo ilustrativo de la diferencia entre imágenes de pixeles y de vectores es el siguiente: Hay dos maneras de enviar una tarta de cumpleaños: Enviarla ya cocinada en una caja, o enviar la receta, de manera que el destinatario la pueda cocinar siguiendo los pasos contenidos en ella

Cuando yo envío un archivo de mapa de pixeles estoy enviando **la tarta entera**. Cuando envío un archivo de vectores, estoy enviando **la receta**, pues los vectores siempre se pueden "cocinar" para obtener un mapa de pixeles.

Tipos

Un pixel puede requerir mayor o menor cantidad de memoria para ser almacenado, y de acuerdo a este valor (llamado la *profundidad de un pixel*) la imagen podrá desplegar una mayor o menor cantidad de colores.

Existen diferentes tipos de vectores o, lo que es igual, diferentes métodos matemáticos de describir una imagen. Por ejemplo, una curva es un primitivo importante de la información vectorial.

Aplicaciones

Utilizados en software de captura, retoque o composición de imágenes reales (video o imagen fija)	Son utilizados generalmente en los programas de dibujo técnico, o modelamiento tridimensional.
- Photoshop - AfterEffects - Premiere	- Illustrator, Freehand, CorelDraw - Flash - 3D Studio, InfiniD, Maya...
La conversión de una imagen de pixeles a una imagen vector se llama *vectorización*.	La conversión de una imagen de vectores a una imagen de pixeles se llama render

Formatos

Los formatos de imágenes de pixeles no dependen tanto de la aplicación con la que fueron creados. Casi todas las aplicaciones que procesan imágenes pixeles pueden leer diversos formatos.	Los formatos de vectores están muy ligados al tipo de software que se utiliza para crearlos o interpretarlos, y aunque existen maneras de convertir de un formato a otro, siempre existe pérdida de información en dicha conversión.
BMP (windows), PSD (photoshop), JPEG, GIF, TIF, TGA.	AI (illustrator), CDR (coreldraw), DXF (autocad)

Existen formatos que almacenan información vector y de pixeles en un sólo archivo. Algunos formatos que pueden contener información mezclada son: PICT (macintosh), WMF (windows), EPS (encapsulated postscript, empleado para impresión en papel) y PDF (formato portátil de adobe)

Ventajas y desventajas

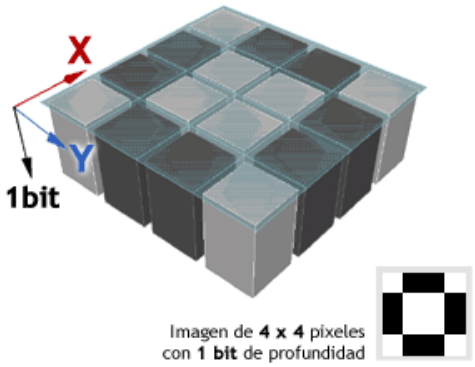
Los pixeles requieren menos operaciones del procesador para ser decodificados.	Requieren mayor cantidad de operaciones del procesador para ser decodificados y desplegados en la pantalla, ya que siempre se convierten finalmente en una imagen de pixeles a través de un proceso de render
Generalmente, por almacenar cada punto de la imagen, ocupan mayor espacio en memoria, y requieren un tiempo mayor de transferencia a través de las redes.	Almacenan en pocos bytes información compleja, de manera que se transfieren rápidamente a través de las redes.

Tienen una resolución fija, determinada por la cantidad de pixeles que se hayan almacenado en el archivo. Cualquier operación de reducción o ampliación de la cantidad de pixeles, redunda en una pérdida de información o aliasing	Son independientes de la resolución, es decir, con la descripción geométrica almacenada se pueden generar imágenes de diversos tamaños de pixeles, tan sólo ampliando la escala del vector.
Son buenos para almacenar texturas complejas.	No son buenos para almacenar texturas, sino más bien áreas de color plano.

Algunos conceptos importantes son los siguientes:

Profundidad de color

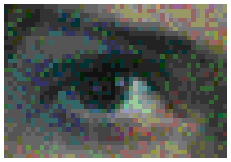
Si las coordenadas del pixel determinan su posición en la imagen, la profundidad es la cantidad de memoria requerida para almacenar su color.



La profundidad de un pixel **no se debe confundir** con la posición de ese pixel en un eje Z imaginario (considerando los ejes X y Y como su posición en el plano). Esta "profundidad" sólo representa cantidad de información, no posición espacial.

La unidad mínima de almacenamiento en la memoria de un computador es 1 bit, el cual puede tomar solamente dos valores: **1** ó **0**. Por ello, los computadores, en lugar de usar el sistema decimal de numeración que utilizamos en la vida cotidiana, utilizan el sistema binario.

Esto quiere decir que para calcular la cantidad de colores que puede contener una imagen de pixeles, debemos elevar el número **2** a la cantidad de bits utilizados para almacenar el color en un pixel. Los ejemplos de esta fórmula se encuentran aquí abajo.

Imagen	Profundidad	Cantidad de colores	Formatos más utilizados
	1 bit	2^2 = 2 colores	GIF, BMP
	8 bits (1 Byte)	2^8= 256 colores	GIF, BMP
	16 bits (2 bytes)	2^{16} = 65536 colores	BMP, TGA, TIF, PSD, PICT
	24 bits (1 byte Rojos 1 byte Azules 1 byte Verdes)	2^{24} = 16'777.216 colores	BMP, TGA, TIF, PSD, PICT, JPG

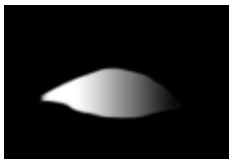
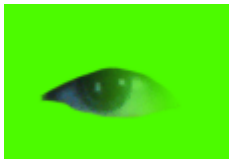
Las imágenes de 8 bits son un caso especial, puesto que su color se define por índices (color "indexado") o números almacenados en una tabla de color.

La diferencia entre una imagen de 16 y 24 bits sólo es notoria en colores suavemente degradados. En la imagen de 16 bits se ven mucho más las bandas de color, debido a la falta de colores para representar un degradado continuo.

16 bits	
24 bits	

Imágenes de 32 bits

Existen además imágenes con profundidad de pixel de 32 bits. Los 8 bits (1 byte) adicionales de profundidad sobre las imágenes de 24 bits, le permiten almacenar la transparencia de la imagen. Este byte adicional es generalmente llamado **máscara** o **canal alfa**, y almacena, en una imagen de 256 niveles de grises, diferentes valores de transparencia.

	Imagen: 24 bits (3 bytes de color)	2^{24} = 16'777.216 colores
	Máscara: 8 bits (1 byte de transparencia)	2^{8} = 256 niveles de transparencia
	Imagen resultante sobre fondo verde	2^{32} = color + transparencia

Normalmente, un pixel blanco en la máscara hace que el pixel correspondiente en la imagen se muestre completamente opaco (no deja ver el fondo) y un pixel negro en la máscara hace al pixel de la imagen completamente transparente (deja ver el fondo). Los grises logran transparencias intermedias.

Tamaño de una imagen de pixeles en la memoria:

Se puede calcular el tamaño de cualquier archivo de imagen de pixeles multiplicando la cantidad de pixeles horizontales por la cantidad de pixeles verticales, y luego multiplicar ese producto por

la profundidad, así:

Tamaño en pixeles	Profundidad de pixel	Tamaño del archivo			
		bits	**bytes**	**Kbytes**	**Mbytes**
640 x 480	x 1 bit	= 307.200	= 38.400	= 37.5	= 0.036
640 x 480	x 8 bits	= 2´457.600	= 307.200	= 300	= 0.292
640 x 480	x 24 bits	= 7´372.800	= 921.600	= 900	= 0.878
640 x 480	x 32 bits	= 9´830.400	= 1´228.800	= 1200	= 1.171

Compresión

La compresión es un término importantísimo en el almacenamiento y transferencia digital de la información. Compresión es cualquier tipo de proceso (o *algoritmo*) que reduzca la cantidad de información contenida en un archivo, ya sea perdiendo o no parte de la información original. Dicha pérdida se denomina aliasing

De hecho, convertir una imagen de 24 bits (un archivo TGA, por ejemplo) a una imagen de 8 bits (un GIF, p.ej.) es un proceso de compresión, puesto que estoy reduciendo la cantidad de memoria que necesito para representar cada pixel de la imagen original.

Una compresión sin pérdidas devuelve la imagen descomprimida exactamente igual a la original. Por el contrario, la compresión con pérdidas acepta alguna degradación en la imagen de cara a una mayor compresión.

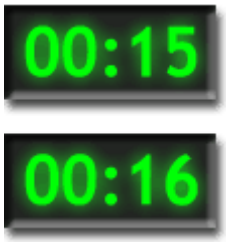
Aliasing

Aliasing es la palabra acuñada por la teoría de la información para denotar la pérdida de datos después de un proceso o transferencia de información. En la imagen digital el aliasing redunda en la pérdida de calidad, en imágenes digitales de poca profundidad de pixel, genera algunas veces el efecto de *escalera* o *línea dentada*, que se ve sobre todo en las líneas curvas u oblicuas.

Representaciones Continuas y Discretas

Supongamos que tenemos que representar el paso del tiempo:

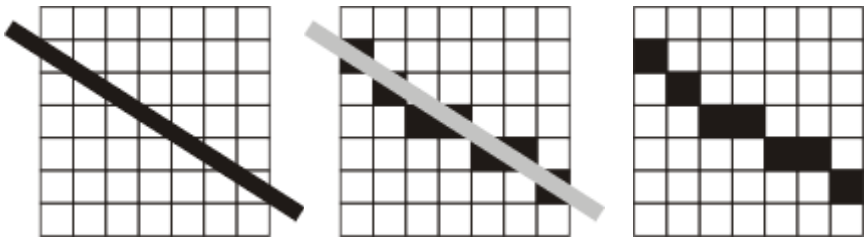
Si tomamos un reloj de agujas (o *analógico*) para hacerlo, vemos que las agujas describen el recorrido circular **continuamente**, es decir, cuando el segundero pasa del segundo 15 al segundo 30, recorre físicamente **todo el espacio disponible** entre una posición y otra..



Por el contrario, cuando tomamos un reloj digital (o *discreto*) que no marca fracciones de segundo, el paso entre el segundo 15 y el segundo 16 nunca se representa. Por ello, es una representación **discontinua**. No importa cuánta aproximación tengamos (supongamos, millonésimas de segundo) siempre se dejará de representar la continuidad real del tiempo.

La rejilla de pixeles

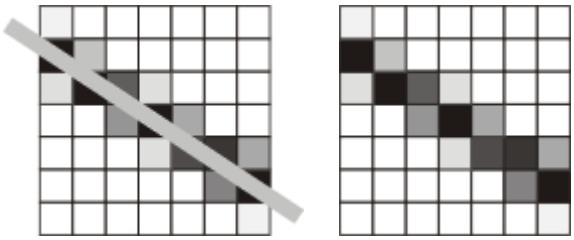
En la imagen digital, por lo menos en los sistemas más utilizados hoy en día, la imagen se divide en una rejilla *discreta* (discontinua) de pixeles, de resolución limitada. Si quisiéramos representar una línea continua con una imagen de 7 x 7 pixeles, ocurriría lo siguiente:



En la imagen de la izquierda, tenemos la línea original. Un pixel sólo puede ser negro o blanco (considerando 1 solo bit de profundidad). En este caso, no se puede representar un pixel cubierto parcialmente por la línea. Debemos, pues, eliminar parte de la información original, para obtener el resultado de la imagen de la derecha. El resultado: una imagen con un aliasing muy alto.

Anti-alias

Supongamos que tenemos una mayor profundidad de pixel para representar la misma línea, digamos 8 bits (256 grises, en este caso). ¿Cómo se puede recuperar la información perdida?



Inventándo un *algoritmo* de antialias, es decir, un proceso que permita recuperar algo de lo perdido, hice que el porcentaje de negro de cada pixel dependiera de qué tan cerca está el centro de la línea del centro del pixel.

Así, al primer pixel de la esquina superior izquierda, cuyo centro está muy lejos del centro de la línea, le asigno un color con porcentaje de negro muy bajo (digamos 5%). Y al pixel central, cuyo centro coincide con el centro de la línea, le asigno un negro total (100%).

Este algoritmo hechizo puede no reflejar lo que hace un verdadero algoritmo de anti-alias, pero arroja un resultado similar a los algoritmos de los software actuales, que se manifiesta como un degradado de colores entre el color del objeto y el color del fondo.
El aliasing ocurre entonces cuando se intenta representar una información **continua** a través de un medio **discreto**. En el caso de las gráficas digitales este paso ocurre cuando convertimos una información vector en una imagen de pixeles proceso llamado **render**, o en cualquier proceso de digitalización, cuando, por ejemplo, se escanea, se captura una imagen a través de una cámara de video o se mueve el ratón para dibujar una línea.

Formatos de archivos de imagen.

TIFF *(*Tagged Image File Format*)*

TIFF es, en principio, un formato muy flexible con o sin pérdida. Los detalles del algoritmo de almacenamiento de la imagen se incluyen como parte del fichero. En la práctica, TIFF se usa casi exclusivamente como formato de almacenamiento de imágenes sin pérdidas y sin ninguna compresión. Consecuentemente, los archivos en este formato suelen ser muy grandes. Algunas veces se usa un algoritmo de compresión sin pérdidas llamado LZW, pero no siempre.

El formato TIF (Formato de archivo de imágenes con etiquetas) es un formato de archivo de gráficos de mapa de bits (una trama). Fue desarrollado en 1987 por Aldus (ahora pertenece a Adobe). Las últimas especificaciones (Revisión 6.0) se publicaron en 1992.

Además, el formato TIF permite que se utilicen varios espacios de color: RGB (rojo, verde, azul) CMYK (cian, magenta, amarillo, negro) etc.

El principio del formato TIF consiste en definir etiquetas (de ahí el nombre Formato de archivo de imágenes con etiquetas) que describen las características de la imagen. Las etiquetas permiten almacenar información acerca de las dimensiones de la imagen, la cantidad de colores utilizados, el tipo de compresión, etc. Por lo tanto, una descripción de imagen que utiliza etiquetas simplifica la programación del software permitiendo guardar información en formato TIFF. Por otro lado, la cantidad de opciones es tan amplia que muchos editores de imágenes que admiten el formato TIFF no las integran todas.

Otra ventaja de este formato que, por su importancia, merece ser reseñada, es su implantación. Y es que los ficheros TIFF **pueden ser manipulados por prácticamente cualquier aplicación de edición fotográfica o diseño** del mercado. Y esto no es todo. Además, pueden almacenarse con capas y guardarse una y otra vez sin deteriorarse.

A pesar de todo, la en su momento ingeniosa tecnología de Aldus ha sido batida. Actualmente tenemos a nuestra disposición otros formatos de imagen con características más atractivas que TIFF. Los ficheros almacenados en este último formato **son pesados**, sobre todo si contienen varias capas, y su compresión no es muy eficiente. Hay opciones, como **PSD**, por ejemplo, que ofrecen una **calidad equivalente** y una **compresión mucho mayor que TIFF**.

Además, las técnicas de compresión que se aplican actualmente a los archivos JPEG se han depurado mucho. De hecho, un JPEG puede ofrecer un **acabado cercano al de un fichero comprimido sin pérdida de calidad**, pero **ofreciendo un «peso» mucho menor**. Aunque, si guardamos muchas veces el mismo archivo, la pérdida de calidad será cada vez más perceptible.

GIF(*Graphic Interchange Format, Formato de intercambio de gráficos)*

GIF crea una tabla de 256 colores a partir de una de 16 millones. *Es un formato de archivos de gráficos de mapa de bits (una trama) desarrollado por Compuserve.*

Si la imagen tiene menos de 256 colores, GIF puede almacenar la imagen sin pérdidas. Cuando la imagen contiene muchos colores, el software que crea el archivo GIF usa algún algoritmo para aproximar los colores de la imagen con una paleta limitada de 256 colores disponibles. Un buen algoritmo de este tipo, tratará de encontrar un conjunto óptimo de 256 colores. Algunas veces, GIF usa el color más cercano para representar cada píxel, y algunas veces usa un "error de difusión" para ajustar los colores de los píxeles vecinos y así corregir el error producido en cada píxel.

GIF produce compresión de dos formas. Primero, reduce el número de colores de la imagen a 256 y por tanto, reduce el número de bits necesario por píxel. Después, remplaza áreas de color uniforme usando código de secuencias: en lugar de almacenar "blanco, blanco, blanco, blanco, blanco" almacena "5 blanco"

Por tanto, GIF es una compresión de imágenes sin pérdida sólo para imágenes de 256 colores o menos. Sin embargo, para una imagen de 16 millones de colores GIF puede "perder" el 99.998% de los colores.

Dado que la paleta tiene un número de colores limitado (no limitado en cuanto a colores diferentes), las imágenes que se obtenían con este formato por lo general eran muy pequeñas.

Sin embargo, dado que el algoritmo de compresión LZW estaba patentado, todos los editores de software que usaban imágenes GIF debían pagarle regalías a Unisys, la compañía propietaria de los derechos. Esta es una de las razones por las que el formato PNG se está volviendo cada vez más popular, en perjuicio del formato GIF.

El uso de los GIF es usado generalmente para la publicidad en tipo banners. Su principal utilidad hoy en día sigue siendo el despliegue de imágenes animadas para páginas web, al ser el único formato soportado por multitud de navegadores que permita dicho efecto. Cabe destacar que la animación de este tipo de imágenes solo se puede visualizar en cierto tipo de aplicaciones y programas como presentaciones power point o páginas web, pero en hojas de cálculo o documentos de texto las imágenes gif pierden su animación.

Las redes sociales han provocado una nueva edad de oro en este formato,[2] que había perdido terreno, frente a otros de alta resolución para las fotografías. Las redes sociales como Google Plus o Tumblr que permiten las animaciones han hecho que el gif animado vuelva a ser un formato muy utilizado por su sencillez de edición y poco peso frente a los vídeos.

Como juntar imágenes para hacer a GIF animado.

Existen generadores de GIF animados en línea a partir de un conjunto de imágenes fijas como Gickr (permite juntar hasta 10 fotos, escoger la dimensión en píxeles y la velocidad del GIF) y Makeagif (permite hacer la misma cosa, pero también generar y compartir directamente la URL del GIF).

Es también posible realizar un GIF animado a partir de programas gratuitos como Gimp o Unfreez.

En el programa o sitio web, escoge imágenes del mismo tamaño, copia la segunda imagen en la primera, la tercera en la segunda y así sucesivamente en el orden de aparición deseado. Escoge la velocidad de secuencia de las imágenes y los efectos que hay que aplicar en la medida de lo posible. No olvides guardar la imagen en formato **GIF**.

Cómo hacer un GIF animado a partir de un vídeo

Como con las fotos, basta con importar un vídeo para crear un GIF y seleccionar el paso que se

desea transformar. Puedes realizar un GIF animado a partir de un vídeo desde sitios web como Bloggif (puedes importar vídeos de 30 MB como máximo) o Gifsoup, que te permite realizar un GIF animado a partir de una URL de YouTube.

Ciertos programas como Format Factory o QGifer te proponen también crear un GIF animado a partir de un vídeo.

Para crear un archivo GIF desde un vídeo: importa o indica la URL del vídeo deseado, determina el principio y el final de la secuencia, escoge las opciones como el tamaño, los filtros de color, la secuencia de imágenes, previsualiza el GiF y guárdalo en formato GIF.

*PNG(*Gráficos de Red Portátiles*)*

PNG es también un formato de almacenamiento sin pérdida. Puede comprimir la imagen. Además tal compresión es totalmente reversible y por tanto la imagen que se recupera es exacta a la original.

Fue desarrollado en 1995 como una alternativa gratuita al formato GIF, que es un formato patentado cuyos derechos pertenecen a Unisys (propietario del algoritmo de compresión LZW), a quien todos los editores de software que usan este tipo de formato deben pagar regalías. Por lo tanto, PGN es un acrónimo recursivo de *PNG No es GIF*.

El formato PNG permite almacenar imágenes en blanco y negro (una profundidad de color de 16 bits por píxel) y en *color real* (una profundidad de color de 48 bits por píxel), así como también imágenes indexadas, utilizando una paleta de 256 colores.

Además, soporta la transparencia de canal alfa, es decir, la posibilidad de definir 256 niveles de transparencia, mientras que el formato GIF permite que se defina como transparente sólo un color de la paleta. También posee una función de entrelazado que permite mostrar la imagen de forma gradual.

La compresión que ofrece este formato es (*compresión sin pérdida*) de 5 a 25% mejor que la compresión GIF.

Por último, el PNG almacena información gama de la imagen, que posibilita una corrección de

gama y permite que sea independiente del dispositivo de visualización. Los mecanismos de corrección de errores también están almacenados en el archivo para garantizar la integridad.

JPG

JPG es el método de compresión más adecuado para fotografías e imágenes de tonos continuos similares que contiene muchos colores. Permite obtener unos radios de compresión muy altos manteniendo a su vez una calidad en la imagen muy elevada. JPG analiza las imágenes y elimina la información que no es apreciable. JPG almacena imágenes de 16 millones de colores. Otro aspecto importante es que el método JPG permite distintos niveles de compresión. En niveles de compresión de imágenes moderado, es muy difícil discernir las diferencias de la imagen original. Programas de tratamiento de imágenes avanzados como Paint Shop Pro o Photoshop permiten ver la calidad de la imagen y el tamaño del fichero como una función de nivel de compresión, de esa forma, se puede elegir convenientemente la calidad y el tamaño del fichero deseado.

SVG (Scalable Vector Graphics) es un formato vectorial poco conocido pero muy útil para su uso online por su flexibilidad y por la capacidad de ofrecer gráficos con calidad.

El formato SVG es para muchos un total desconocido. Cuando queremos colocar algún gráfico en la web, en la mayoría de ocasiones optamos por JPG o alguna vez por un PNG o GIF, en caso de necesitar transparencia o animación. Lo que muchos no saben es que se pueden usar archivos vectoriales para su uso en navegadores.

VECTORIAL

SVG es vectorial, lo que supone tener todas las ventajas de cualquier formato vectorial. Es escalable, pesa poco y permite una definición mayor a tamaños reducidos, mucho mayor que los archivos bitmap. El formato es igual al que se utiliza con cualquier programa vectorial como Corel Draw o Adobe Illustrator.

STANDARD ABIERTO Y COMPATIBLE

Así como el formato Flash, que también era vectorial, propiedad de Macromedia, es decir de Adobe, el formato SVG es un formato abierto, estándar y basado en XML. Aunque las primeras versiones no se podían ver en los diferentes navegadores, hoy ya es un estándar que funciona sin problemas en todos los navegadores. SVG se convirtió en una recomendación del W3C en septiembre de 2001 con lo que en estos momentos ya es admitido por todos. A ser un formato basado en XML necesitamos cierto control de código para hacer que un archivo SVG funcione adecuadamente.

TIPOGRAFÍAS

El formato SVG nos permite por un lado utilizar las tipografías con trazados pero también nos permite incluirlas dentro del propio archivo en formato TrueType y Tipo 1, lo que nos da una capacidad extraordinaria en cuanto a que los motores de búsqueda son capaces de indexarlo. Hay que tener en cuenta que para el texto puro colocado como tipografía tiene que ser o bien de las fuentes instaladas en el sistema o bien incluidas como estilo CSS.

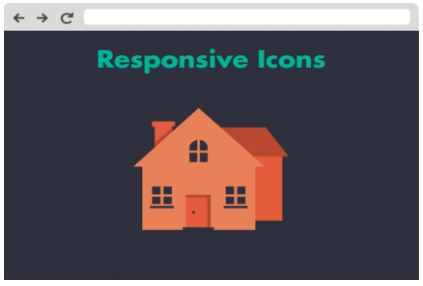
Proyectos como Iconic también usan el formato SVG para hacer que una tipografía con iconos nos sirva para los diferentes usos y necesidades de tamaño y variación en función de las acciones que deseemos. Puedes ver ejemplos en su web con los que se entenderás realmente y en profundidad el verdadero uso del formato SVG.

INTERACTIVO

Los archivos SVG pesan poco, igual que lo haría un archivo vectorial. Soportan estilos CSS, con lo que si cambiamos el estilo en nuestra web también cambiará dentro del archivo. Puede incluir scripts que permiten caminos dinámicos. Puede admitir acciones como los *rollovers* o cambios al hacer click. Puedes ver un ejemplo en este enlace: http://codepen.io/chriscoyier/pen/evcBu

ICONOS RESPONSIVE

En la actualidad todas las webs están migrando al formato *responsive,* así que nos encontramos multitud de iconos que se redimensionan en función del ancho de la pantalla. Normalmente el funcionamiento suele ser sustituir un icono bitmap por otro más pequeño. Con SVG podemos utilizar el mismo y que se vaya reduciendo poco a poco, o bien que utilizar varios tamaños pero que todos ellos sean SVG con la ventaja de la mejor visualización al ser vectorial.



RAW, BMP, PSP, PSD, ...

RAW es la imagen de salida que ofrece algunas cámaras digitales. Aunque es un método sin pérdida, ofrece un factor de tres o cuatro menor que el formato TIFF de la misma imagen. La desventaja es que el método RAW no está estandarizado y cada marca tiene su propia versión de dicho método, por tanto, se debe usar el software de la cámara para poder visualizar las imágenes.

BMP es un formato de almacenamiento sin compresión de imágenes propiedad de Microsoft.

PSP, PSD son formatos usados en distintos programas básicos (Paint Shop Pro, Photoshop).

En resumen, los métodos de compresión TIFF, PNG, GIF y JPG son públicos y por tanto se pueden implementar en cualquier programa gráfico. GIF y JPG son los más usados en las páginas web. Dado que PNG hace lo mismo que GIF e incluso mejor, se espera que PNG sustituirá GIF en el futuro. PNG no sustituirá JPG dado que JPG consigue una mayor compresión en imágenes fotográficas.

Comparando tamaño de ficheros

Tipo de fichero	Tamaño	Ejemplo
TIFF sin ningún tipo de compresión	*901K*	*Clickhereto try.*
TIFF con compresión LZW	*928K*	*Clickhereto try.*
JPG	*105K*	*Clickhere.*
PNG, compresión sin pérdida	*741K*	*Clickhere.*
GIF, compresión sin pérdida (256 colores)	*131K*	*Clickhere.*

¿Cuál usar?

TIFF

Normalmente, es el formato de mayor calidad en una cámara digital. Las cámaras digitales ofrecen alrededor de tres niveles de compresión JPG aparte de TIFF. De todas formas, el tamaño del fichero TIFF es mucho mayor que el de JPG de mejor calidad y la diferencia no es apreciable.

Un uso importante del formato TIFF es editar y manipular imágenes, ya que si trabajamos directamente con el formato JPG, se va acumulando los errores cada vez que grabamos la imagen.

JPG (Joint Photograph Experts Group)

Este es el formato más elegido en las fotografías en la web, dado que produce una excelente calidad incluso con radio de compresión muy elevados.

Sin embargo, nunca se debe usar el formato JPG para editar imágenes a trazos o con líneas delgadas, ya que en imágenes con áreas de color uniforme, JPG produce muchos errores. En este caso se debe usar el formato GIF o PNG.

Además, al ser JPG un método de compresión con pérdidas, no se debe manipular una imagen directamente en JPG e ir almacenando pues los errores se van acumulando.

GIF (Graphic Interchange Format)

Si la imagen contiene menos de 256 colores y grandes áreas de color uniforme, GIF es el mejor formato para guardarla. Sin embargo, no se deben usar GIF para imágenes fotográficas ya que sólo almacena 256 colores por imagen.

PNG (Portable Network Graphics)

Las propiedades más importantes de PNG son:

1. *Sirve para comprimir sin pérdida imágenes con grandes áreas de color uniforme, y con más de 256 colores. PNG es similar a GIF con la salvedad de que es capaz de almacenar 16 millones de colores y no sólo 256.*
2. *Es útil si se quiere mostrar una imagen fotográfica exactamente en la web. La otra opción sería TIFF, pero algunos navegadores no admiten este formato.*

Actualmente el formato GIF es más usado en la web que PNG, dado que los navegadores más antiguos soportan GIF pero no PNG.

SOFTWARE PARA CREAR Y PROCESAR IMÁGENES

Para la creación, manipulación y tratamiento de imágenes existe una gran cantidad de oferta de herramientas, tanto comerciales como gratuitas, algunas de estas últimas se ofrecen a través de la Web. Seguidamente se identifican algunas de ellas:

Una de las herramientas comerciales más conocidas para el tratamiento digital de imágenes, de hecho algunas personas han llegado a asociar retoque de imágenes y esta herramienta, es Adobe Photoshop, Photoshop es un producto software para la creación, edición y retoque de imágenes. Los formatos propios de Photoshop son PSD y PDD, que guardan capas, canales, guías y cualquier modo de color. Además de los formatos comentados también soporta la mayoría de tipos de imágenes comentados en secciones anteriores.

Como alternativa en el ámbito de software libre a Photoshop está GIMP. GIMP*(GNU ImageManipulationProgram)* es un programa de edición de imágenes digitales en distintos formatos. Es un programa

libre y gratuito. Está englobado en el proyecto GNU y disponible bajo la Licencia pública general de GNU

Algunas herramientas software para manipular, optimizar y crear imágenes de acceso a través de la Web son las siguientes:

- **Pixlr**: es un editor de imágenes *on line,* fácil de manejar y permite realizar las operaciones básicas de tratamiento de imágenes. Tiene menor potencia que Photoshop o GIMP, pero es más fácil de manejar.
- **Cellsea Free Web Photo Editor**: Cellsea es un editor de imágenes y fotografía *on line* bastante completo. Tiene todo tipo de herramientas y, en lo que a retoque de imágenes se refiere, poco tiene que envidiarle a herramientas comerciales como Photoshop.
- **Fauxto**: otra herramienta muy recomendable para dar soporte a la edición de fotos e imágenes *on line.* ofrece a través de una interfaz de usuario casi igual a la de Photoshop y tiene las mismas herramientas, permite utilizar capas *(layers)* y filtros. Es muy usada por profesionales y no es tan simple como la anterior herramienta.
- I**mageTool**: herramienta para crear logos *on line.* El usuario tiene que elegir el color de fondo y las propiedades del texto. Con esos datos es suficiente.

OPTIMIZACIÓN DE IMÁGENES PARA LA WEB. RESOLUCIÓN

Al crear un sitio web es muy recomendable que los archivos que contienen las imágenes ocupen el menor número posible de bytes para agilizar su descarga y visualización por Internet. Esto garantizará el acceso de aquellos clientes que utilicen conexiones con anchos de banda modestos.

El tamaño de un archivo gráfico viene determinado, entre otros, por los siguientes factores:

- Dimensiones de la imagen.
- Profundidad o paleta de colores.
- Resolución.
- Formato de archivo (JPG, GIF, PNG).

Recomendaciones de optimización

Conviene definir una resolución de imagen no superior a 96 ppp. Es la resolución que usan las pantallas de ordenador. No interesa optar por valores mayores, ya que aumenta considerablemente el peso del archivo a descargary el usuario no lo aprecia.

En ocasiones puede interesar reducir el número de colores de la paleta porque ello supone disminuir el tamaño del archivo.

Conviene utilizar un programa de tratamiento de imágenes para definir las dimensiones concretas de una imagen antes de insertarla en una página web.

Es recomendable guardar los originales de las imágenes favoritas en

formato BMP, TIFF ó JPEG sin comprimir. A partir de ellas se puede crear una copia en formato GIF (PNG) o JPEG con las dimensiones, resolución y paletas optimizadas para publicarlas en la web.

Las imágenes GIF son más adecuadas para dibujos, gráficos y logotipos. Son aquellas donde predominan los colores sólidos y una paleta con un número reducido de colores.

Las imágenes JPEG se adaptan mejor a fotografías e imágenes con degradados complejos. Admiten color de 24 bits gracias a su compresión ofrecen una imagen más brillante que ocupa menos espacio.

La correcta combinación de aspectos relacionados con la resolución, el color, el número de bits y el formato, es muy importante para la optimización de una imagen. Puesto que en la web se deben cumplir, por encima de todo, dos principios: que las imágenes se vean bien y que pesen poco, para que no tarden mucho en cargarse.

Herramientas de optimización

La mayoría de las herramientas consideradas en la sección anterior nos permiten optimizar las imágenes para utilizarlas en la Web, pero hay herramientas específicas y relacionadas con actividades de optimización y reducción del tamaño, por ejemplo:

ImageOptimízatíon: con esta herramienta *on line* se puede cargar una imagen de mapa de bits (GIF, JPG o PNG) y optimizarla para su uso en la Web, sin sacrificar calidad.

DoSize: permite cambiar el tamaño de una imagen y adaptarlo a las necesidades del diseñador.

El nuevo elemento <picture> de HTML5 para crear imágenes responsive

El nuevo elemento <picture> de HTML5 permite describir con todo detalle cómo deben cargarse las imágenes de tu sitio web. Ya no serán necesarios los *hacks* de CSS o JavaScript para gestionar las imágenes responsive de los diseños web. Además, los usuarios se aprovecharán de las ventajas de cargar solamente las imágenes optimizadas para el dispositivo que están utilizando, lo que es especialmente útil para usuarios con móviles y conexiones lentas a Internet.

Al margen de los nuevos atributos srcset y sizes definidos recientemente para los elementos <img>, el nuevo elemento <picture> permite una mayor flexibilidad al especificar qué imágenes utiliza el sitio. Gracias a este elemento <picture>, será posible escribir código HTML *limpio* y semántico, dejando que el navegador haga todo el trabajo de seleccionar la mejor imagen para cada situación.

La elección del mejor archivo de imagen depende de muchos factores:

Elección basada en el diseño gráfico

¿El dispositivo es un móvil en vertical o es un monitor panorámico? Carga la mejor imagen optimizada para el tamaño de la pantalla.

Elección basada en la densidad de píxeles

¿Se trata de un dispositivo de alta resolución? Carga las imágenes de alta resolución.

Elección basada en cómo se visualizará la imagen

¿La imagen debe ocupar siempre un tamaño determinado de la ventana del navegador? Carga las imágenes en función del tamaño de la ventana del navegador.

Elección basada en el formato de la imagen

¿Soporta el navegador formatos de imagen con mucha mayor compresión que los tradicionales?Carga un formato de imagen alternativo, como por ejemplo WebP.

Seleccionando la imagen en función de criterios artísticos

El uso más habitual del elemento <picture> consiste en elegir la mejor imagen exclusivamente en función de criterios artísticos. En vez de diseñar una única imagen que se escala para ajustarse al tamaño de la ventana del navegador, se pueden diseñar diferentes imágenes en función de su tamaño.

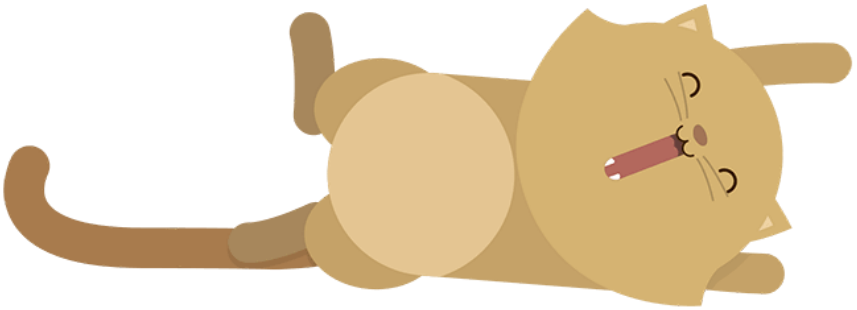
Izquierda: la misma imagen se escala para todos los tamaños de ventana. **Derecha:** diferentes imágenes en función del tamaño de la ventana del navegador.

Mejorando el rendimiento al cargar las imágenes

Cuando se utiliza el nuevo elemento <picture> o el elemento <img> con los atributos srcset y sizes, el navegador solamente descarga la imagen adecuada para el navegador y las condiciones de acceso del usuario (tamaño del navegador, densidad en píxeles de la pantalla, formatos soportados por el navegador, etc.) La ventaja de que este comportamiento sea nativo del navegador es que se pueden aprovechar todas las funcionalidades de los navegadores, como la caché de contenidos y la precarga de imágenes.

El elemento en acción

Como sabes, Internet se inventó para mostrar fotos de gatitos, así que vamos a utilizar el elemento<picture> en acción mostrando cómo se ajusta nuestro gato al espacio disponible en el navegador.



Abrir la demo en una nueva pestaña del navegador. Para ver el elemento `<picture>` en acción, asegúrate de utilizar el navegador Chrome 38 y redimensiona la ventana del navegador para observar cómo cambia la imagen mostrada.

Esta demo es muy básica porque es una primera toma de contacto con las posibilidades del nuevo elemento `<picture>`. Sigue leyendo para conocer todas sus posibilidades.

La sintaxis del elemento <picture>

El siguiente código HTML y CSS muestra todo lo necesario para crear la anterior demo:

```
<style>
    img {display: block; margin: 0 auto;}
</style>
<picture>
  <source
  media="(min-width: 650px)"
  srcset="images/kitten-stretching.png">
  <source
  media="(min-width: 465px)"
  srcset="images/kitten-sitting.png">
  <img
  src="images/kitten-curled.png"
  alt="a cute kitten">

</picture>
```

Como puedes ver, no se utiliza ni código JavaScript ni ninguna otra librería externa. El bloque de código CSS se utiliza para aplicar unos estilos básicos a la imagen y de nuevo puedes ver que no se utilizan *hacks* ni *media queries*. Cuando el navegador soporta el elemento `<picture>`, tu único trabajo consiste en definir todas las imágenes *responsive* que tienes disponibles y es el navegador el que se encarga de seleccionar la mejor alternativa.

Uso de <picture> con los elementos <source>

El elemento <picture> no define ningún atributo propio, pero puedes conseguir comportamientos muy avanzados cuando utilizas <picture> para encerrar a varios elementos <source>.

El elemento <source>, que se utiliza para cargar elementos multimedia como audios y vídeos, se ha actualizado para que también soporte la carga de imágenes. Para ello, se le han añadido los siguientes atributos:

Atributo srcset (obligatorio)

Indica la ruta de la imagen a la que se hace referencia (ejemplo srcset="kitten.png").

También se puede indicar una lista de rutas separadas por comas y que incluyan el sufijo que indica la densidad de píxeles (ejemplo srcset="kitten.png, kitten@2X.png 2x"). Para la densidad de píxeles normales de 1 no hace falta añadir el descriptor 1x.

Lee la sección *Seleccionando la imagen en función de la densidad de píxeles* para saber cómo utilizarlo en la práctica.

Atributo media (opcional)

Permite indicar cualquier *media query* que sea válido en el selector @media de CSS (ejemplo media="(max-width: 30em)").

Atributo sizes (opcional)

Acepta cualquier valor que describa la anchura de la imagen (ejemplo sizes="100vw") o un *media query* que defina la anchura de la imagen (ejemplo sizes="(max-width: 30em) 100vw").

También se puede indicar una lista ed *media queries* separadas por comas y que describan varias anchuras de la imagen (ejemplo sizes="(max-width: 30em) 100vw, (max-width: 50em) 50vw, calc(33vw - 100px)"). En este caso se utiliza por defecto el último de los valores definidos.

Atributo type (opcional)

Acepta como valor cualquier tipo MIME estándar (ejemplo type="image/webp" o type="image/vnd.ms-photo").

Lee la sección *Seleccionando diferentes formatos de imagen* para saber cómo utilizarlo en la práctica.

El navegador utiliza el valor de todos los atributos anteriores para determinar qué imagen se debe cargar de entre todas las variantes definidas. Ten en cuenta que **el orden de las etiquetas es muy importante**, ya que el navegador siempre utilizará el primer elemento <source> cuyas condiciones cumpla el navegador e ignorará el resto de elementos <source>.

Añade un elemento <img> al final

El elemento <img> también se ha actualizado para poder utilizarlo dentro del elemento <picture> a modo de salvaguarda en el caso de que el navegador no soporte <picture> o ninguna de las condiciones de los elementos <source> se cumplan.

Añadir un elemento <img> dentro del elemento <picture> **es obligatorio**. Si no lo haces, **el navegador no mostrará ninguna imagen**.

La imagen definida por el elemento <img> será la que utilizará el elemento <picture> por defecto cuando no se puede mostrar ninguna otra de las imágenes definidas. Coloca el elemento <img> como último elemento hijo de <picture>, ya que los navegadores ignoran cualquier elemento <source> que se encuentre después de la etiqueta <img>. Si defines un texto alternativo para la imagen mediante el atributo alt, asegúrate de añadir ese atributo en la etiqueta <img>, no en <source> o <picture>.

Seleccionando la imagen en función de la densidad de píxeles

Utiliza los descriptores 1x, 1.5x, 2x y 3x para añadir soporte para pantallas de alta densidad de píxeles, como por ejemplo las de los *smartphones*. El atributo srcset que permite indicar estos descriptores ahora se soporta tanto en el elemento <img> como en los elementos <source>.

El siguiente ejemplo muestra cómo soportar las pantallas de tipo 1x, 1.5x y 2.x:

```
<picture>
<source
media="(min-width: 650px)"
srcset="images/kitten-stretching.png,
images/kitten-stretching@1.5x.png 1.5x,
images/kitten-stretching@2x.png 2x">
<source
media="(min-width: 465px)"
srcset="images/kitten-sitting.png,
images/kitten-sitting@1.5x.png 1.5x
images/kitten-sitting@2x.png 2x">
<img
src="images/kitten-curled.png"
srcset="images/kitten-curled@1.5x.png 1.5x,
images/kitten-curled@2x.png 2x"
alt="a cute kitten">
</picture>
```

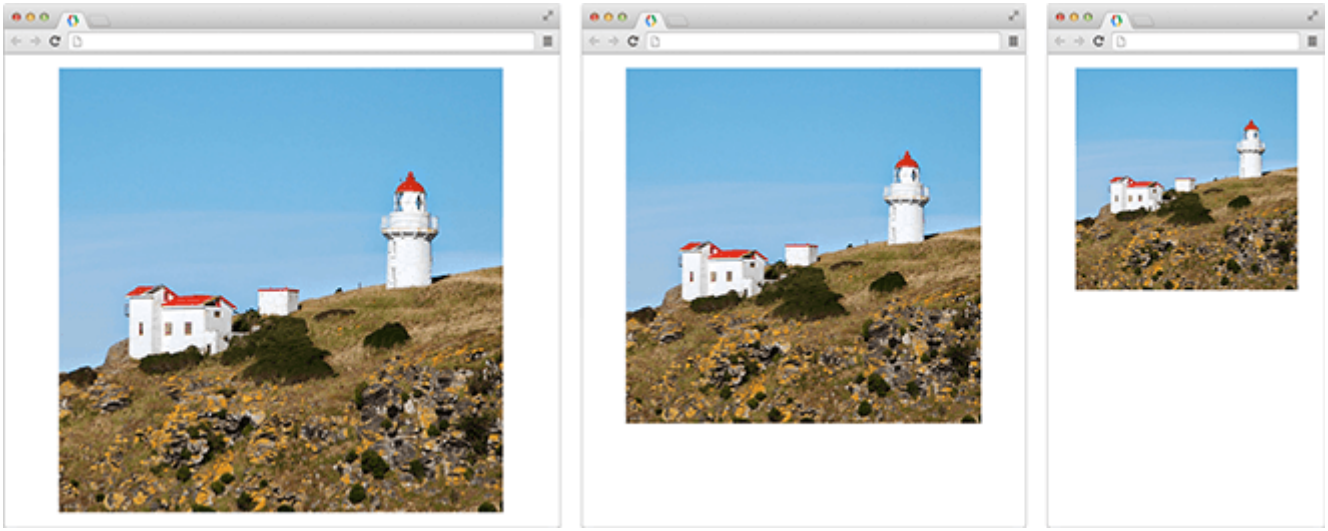
Seleccionando la imagen en función de su anchura

Cuando se desconoce el tamaño definitivo de una imagen, no es posible indicar el descriptor relacionado con la densidad de píxeles mencionado en la sección anterior. Así que en vez de definir imágenes de anchura fija, se puede añadir un descriptor de su anchura para que el navegador calcule automáticamente la densidad de píxeles y así descargue la mejor imagen en cada caso.

En este ejemplo se utiliza el atributo `sizes` para definir que la imagen siempre ocupe el 80% de la anchura de la ventana del navegador. Además, se combina con el atributo `srcset` para definir cuatro versiones diferentes de la misma foto de un faro, cada una con una anchura específica: 160px, 320px, 640px y 1280px:

```
<img src="lighthouse-160.jpg"alt="lighthouse"
sizes="80vw"
srcset="lighthouse-160.jpg 160w,
lighthouse-320.jpg 320w,
lighthouse-640.jpg 640w,
        lighthouse-1280.jpg 1280w">
```

El navegador utiliza este descriptor de la anchura para elegir la mejor imagen en función de la anchura del navegador y de la resolución de la pantalla:



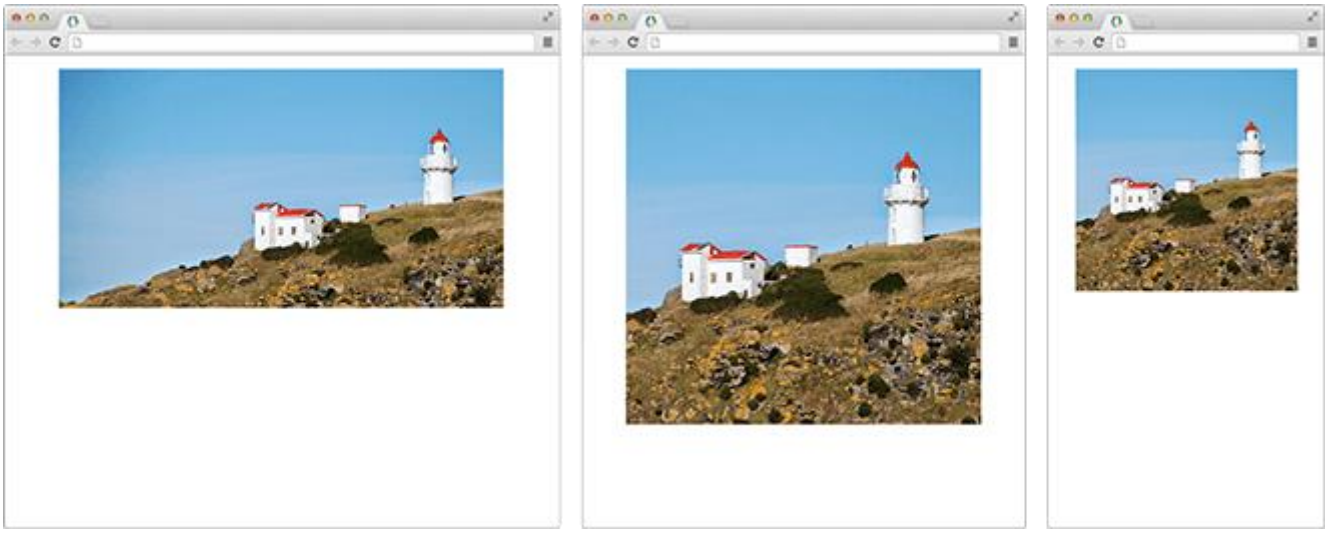
En este ejemplo, la ventana de la izquierda tiene aproximadamente 800px de ancho, por lo que el navegador carga la imagen `lighthouse-640.jpg`. No obstante, si el dispositivo tiene una densidad de píxeles de 2x, entonces se carga la imagen `lighthouse-1280.jpg`.

Al añadir `<picture>`, el atributo `sizes` se puede utilizar tanto en el elemento `<img>` como en los elementos `<source>`:

```
<picture>
<sourcemedia="(min-width: 800px)"
sizes="80vw"
```

```
src set="lighthouse-landscape-640.jpg 640w,
lighthouse-landscape-1280.jpg 1280w,
            lighthouse-landscape-2560.jpg 2560w">
<img src="lighthouse-160.jpg"alt="lighthouse"
sizes="80vw"
srcset="lighthouse-160.jpg 160w,
lighthouse-320.jpg 320w,
lighthouse-640.jpg 640w,
          lighthouse-1280.jpg 1280w">
</picture>
```

Siguiendo con este mismo ejemplo, cuando el navegador tiene una anchura de 800px o superior, se carga la imagen panorámica del faro:



La anchura del navegador de la izquierda es mayor que 800px, por lo que se muestra la versión panorámica de la imagen.

Seleccionando diferentes formatos de imagen

El atributo `type` del elemento `<source>` se puede utilizar para cargar formatos de imagen alternativos que pueden no estar soportados por el navegador del usuario. Si por ejemplo quieres servir imágenes en formato WebP para los navegadores que lo soportan, pero al mismo tiempo mantener las imágenes JPEG para el resto de navegadores, debes utilizar lo siguiente:

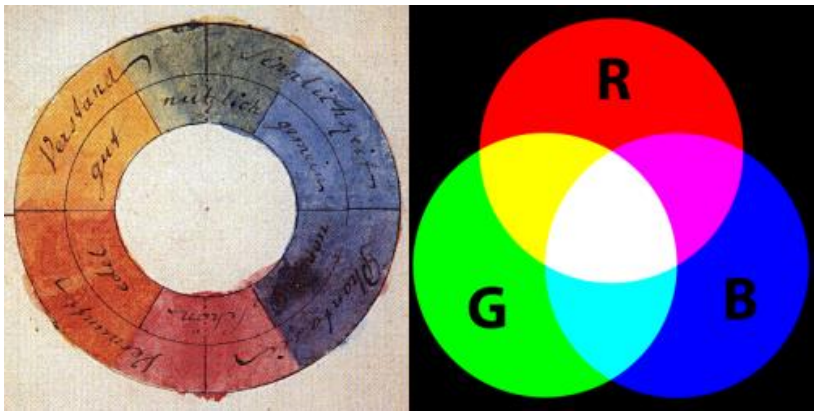
```
<picture>

  <source type="image/webp"srcset="images/butterfly.webp">

  <img src="images/butterfly.jpg"alt="a butterfly">

</picture>
```

RGBA: Los colores transparentes

Cuando queremos que "algo" tenga un determinado **color**, establecemos ese color utilizando algún código, un valor hexadecimal, un valor expresado como RGB() o una palabra que lo identifica:

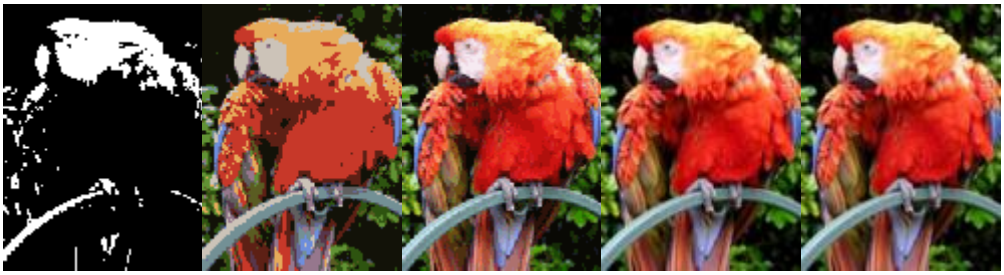
```
color: #FF0000;
color: rgb(255,0,0);
color: rgb(100%,0%,0%);
color: red;
```

Todos ellos harán lo mismo, mostrar algo de color rojo. Es lo que se llama el modelo RGB (Red Green Blue) donde cada color está compuesto por una parte de rojo, una parte de verde y una parte de azul y las diferentes proporciones de estos componentes genera todos los colores posibles.



Como cada componente puede variar entre 0 y 255, tenemos 256 variantes que van desde rgb(0,0,0) o #000000 que genera un color negro hasta rgb(255,255,255) o #FFFFFF que genera un color blanco.

De esa manera tenemos 256*256*256* = 256^3 =16777216 colores diferentes. A medida que la tecnología avanzó, la cantidad de colores disponibles fue aumentando. En algún momento pasaron del "blanco y negro" a soportar 8 colores, luego 16 (4 bits),luego 256 (8 bits), luego 65536 (16 bits) y ahora el llamado *TrueColor* (24 bits)



Cualquiera diría que 16 millones de colores es más que suficiente ya que el ojo humano difícilmente distinga ciertas variaciones sutiles pero, ahora nos enfrentamos con una nueva alternativa, los llamados colores **RGBA** (Red Green Blue Alpha) que agregan un nuevo factor, el *alpha channel*, es decir, la opacidad o transparencia que sigue el mismo esquema de porcentajes: el 0% representa la transparencia absoluta y el 100% representa la opacidad absoluta que es la forma en que tradicionalmente vemos los colores.

Esto es fácilmente visible cuando vemos algunas imágenes en **formato PNG** y no es un avance sin costo ya que pasamos de 24 bits a 32 bits es decir, imágenes más pesadas.

Lo interesante es que el uso de este canal *alpha* no está limitado a las imágenes, también es posible aplicarlo a algunas propiedades CSS aunque sólo está disponible cuando el navegador puede interpretarlo ya que se trata de una posibilidad contemplada en el **CSS3**, algo que en este momento, sólo soportan algunos navegadores.

El valor, al igual que en la propiedad **opacity**, es expresado como porcentaje decimal, yendo de 0 (transparencia total) hasta 1 (el valor por defecto). En este ejemplo, al color rojo se le aplicaron diferentes valores al canal *alpha*:

```
color: rgba(255,0,0,0);
color: rgba(255,0,0,0.2);
color: rgba(255,0,0,0.4);
color: rgba(255,0,0,0.6);
color: rgba(255,0,0,0.8);
color: rgba(255,0,0,1);
```

Funciona correctamente en las versiones más recientes de Filefox 3, Safari, Chrome y Opera 10 pero no así en Internet Explorer, en este navegador, la propiedad será ignorada por lo que no habrá color definido.
Para evitar eso, debemos establecer el color de dos modos, primero, de forma "normal" y luego como **rgba(),** de esta manera, IE usará la primera y el resto de los navegadores la segunda. Por ejemplo:

Sed porta, turpiseuinterdumfacilisis, duiquamvolutpatlorem, euconsequatdui magna velarcu.

```
<div style="background-color:#FF0000;background-color: rgba(255,0,0,0.5);">
 .....
</div>
```

Los valores de **rgba()** pueden ser utilizados en las propiedades **background**, **border** y **color** y, teniendo la precaución de verificar los resultados en los diferentes navegadores, es una buena alternativa, diferente de la propiedad **opacity** ya que sólo afecta a elementos individuales.