

Tema 3: Preparación e implantación de contenido multimedia

¿Qué aprenderás?

- Reconocer los principios básicos de la comunicación visual.
- Crear hojas de estilos seleccionando los colores y tipografías adecuadas a la aplicación.
- Desarrollar wireframes para el diseño de interfaces.
- Maquetar la web siguiendo los wireframes y las hojas de estilos

¿Sabías que...?

- YouTube cada día reproduce unas mil millones de horas de vídeos cada día.
- Más de la mitad de visualizaciones de YouTube provienen de dispositivos móviles.
- John Cage compuso y mantiene los derechos de autor de una obra compuesta por 4 minutos de silencio.



3. Preparación e implantación de contenido multimedia

3.1. Propiedad intelectual

En ésta UF veremos cómo podremos utilizar y modificar contenido multimedia en nuestros sitios web, pero es necesario conocer que no todos los contenidos multimedia que encontremos por la web van a poder ser utilizados libremente.

Los derechos de la propiedad intelectual reconocen el trabajo de los creadores de contenido y busca garantizar que sus obras no serán copiadas, pirateadas o utilizadas sin su consentimiento.

A diferencia de los derechos de propiedad industrial que se adquieren por su registro, el derecho de autor o propiedad intelectual se adquiere cuando se genera la obra para proteger los intereses morales y económicos que la obra pueda representar para él.

Podemos dividir los derechos que conforman la propiedad intelectual en dos grupos:

- **Derechos morales:** Son derechos de carácter personal, irrenunciables e inalienables que acompañan al autor toda su vida. Entre ellos destaca el derecho al reconocimiento de la autoría, el derecho a exigir el respeto a la integridad de la obra y la no alteración de la misma.
- **Derechos económicos:** duran toda la vida del autor más 70 años después de su fallecimiento para sus herederos. Si se trata de una persona jurídica sus derechos de autor durarán 70 años desde la divulgación de la obra. Éstos derechos se pueden vender, ceder o compartir con terceros con interés económico o no.

Existen excepciones o límites a los derechos económicos de autor. Los más importantes son:

- El temporal, transcurridos 70 años desde la muerte del autor la obra pasa a dominio público y puede ser explotada sin permiso del titular de los derechos y sin compensación económica.
- La utilización de una obra para la parodia o para la información.
- Las obras situadas en la vía pública.
- Las entidades de interés general y carácter cultural, científico o educativo sin ánimo de lucro tienen permitido el libre uso, reproducción, préstamos y consulta.
- La copia privada siempre que se haya accedido a ella legalmente y se haga a título personal para uso privado y no comercial.



3.1.1. Utilizar una obra

Cuando queramos poner a disposición del público el original o una copia de una obra estaremos distribuyendo la obra. Para poder distribuir una obra se deberá contar con el permiso del autor o de la obra (o ampararse en alguna excepción).


Si queremos dar acceso a que el público pueda visualizar, escuchar o percibir una obra sin tener acceso directo a ella o a una copia (mediante proyecciones, conferencias, representaciones teatrales, conciertos, etc...) estaremos comunicando públicamente una obra. Para poder comunicar públicamente una obra se deberá contar con el permiso del autor (o ampararse en alguna excepción).

Si queremos utilizar una obra para generar una nueva obra derivada de la original deberemos contar con el permiso del autor (o ampararse en alguna excepción).




3.1.2. Licencias

Las Licencias determinan el uso que el autor autoriza a hacer de su obra y los límites en la responsabilidad derivada de su uso. Las principales licencias que podemos encontrar son:

- Copyright ©: indica que la obra es original e indica la titularidad de los derechos de explotación. Comúnmente precede el lema “todos los derechos reservados” indicando que para su uso, reproducción, transformación o publicación es necesario el permiso del autor.
- Copyleft: una obra con una licencia copyleft, muy utilizada en software, garantiza el derecho de cualquier usuario a utilizar, modificar y redistribuir una obra siempre que comparta las obras derivadas que vaya a crear con una licencia igual o equivalente.
- GNU: es una licencia Copyleft diseñada para manuales o libros de texto que garantizan el derecho de cualquier usuario a utilizar, modificar, redistribuir y vender una obra siempre que la obra siga con la misma licencia, que todos los autores anteriores a la obra sean atribuidos y que todos los cambios que se produzcan en la obra sean registrados.
- Creative Commons: las licencias CC permiten a los autores ceder bajo ciertas condiciones algunos de los derechos sobre sus obras. Con las licencias CC cada autor va a poder combinar las distintas condiciones para generar una licencia a medida que detalle bajo qué condiciones el autor cede distintos derechos sobre su obra.
- Las 4 condiciones básicas que se combinan en las licencias CC son:

- Reconocimiento (BY) : el autor permite requiere que se cite su autoría.



- Uso No comercia (NC) : el autor no permite que se utilice su obra con fines comerciales.
 - Prohibición de obras derivadas (ND) : el autor no permite que no se realicen modificaciones ni se creen obras derivadas.
 - Compartir Igual (SA) : el autor obliga a que las obras derivadas que se creen sean publicadas bajo la misma licencia.
- Cada autor puede combinar las condiciones anteriores para crear una licencia CC ajustada a su medida:



- el autor permite copiar, distribuir y comunicar públicamente la obra mientras se cite su autoría.



- el autor permite copiar, distribuir y comunicar públicamente la obra siempre que no se utilice con fines comerciales y se reconozca su autoría.



- el autor permite copiar, distribuir y comunicar públicamente siempre que no se realicen modificaciones ni se creen obras derivadas y se reconozca su autoría.



- el autor permite copiar, distribuir y comunicar públicamente siempre que las obras derivadas tengan ésta misma licencia y se reconozca su autoría.



- el autor permite copiar, distribuir y comunicar públicamente siempre que no se realicen modificaciones y que las obras derivadas tengan ésta misma licencia y se reconozca su autoría.



- el autor permite copiar, distribuir y comunicar públicamente la obra siempre que no se utilice con fines comerciales ni se realicen modificaciones ni se creen obras derivadas y se reconozca su autoría.

- Creative Commons es una organización fines de lucro dedicada a promover el acceso y el intercambio de cultura que ha desarrollado un conjunto de instrumentos jurídicos de carácter gratuito para usar y compartir el conocimiento.



3.2. Imágenes en la web

Las imágenes son un elemento crucial en todas las páginas web. Las imágenes permiten transmitir emociones al usuario, ilustrar un texto o sintetizar un concepto de forma rápida. Sin embargo para que las imágenes aporten beneficios debemos evitar que el uso de imágenes incremente en exceso el tiempo de espera para cargar una web. Para ello hemos de ajustar lo mejor posibles parámetros de la imagen como su tamaño su resolución o el tipo de compresión.

3.2.1. Resolución

Las imágenes son compuestas por un conjunto de puntos de distintos colores. La resolución de una imagen nos indica cuantos puntos por pulgada se van a usar para mostrar una imagen. A mayor resolución mayor será la calidad de la imagen pero mayor será la cantidad de información necesaria para mostrarla y en consecuencia mayor será su tamaño. Como la mayoría de pantallas utilizan una resolución de 96ppp, ésta es la resolución aconsejada para la mayoría de páginas web, ya que aunque aumentemos su resolución la mayoría de usuarios no van a apreciar un cambio en la calidad.

3.2.2. Tamaños

A demás de la resolución deberemos ajusta los tamaños de la imagen. A mayor anchura y altura de la imagen mayor va a ser su peso. En éste caso debemos ajustar el tamaño de la imagen a la utilización que haremos de la imagen. Si nuestra imagen nunca va a ocupar más de 300px de ancho, no tiene sentido tener la imagen en un tamaño superior, ya que el usuario va a tener que descargarse más información de la necesaria para visualizar la imagen. Si la imagen no tiene un tamaño máximo, deberemos pensar cual será el tamaño que ocupará para una pantalla media de unos 1280 píxeles de ancho.

3.2.3. Profundidad

También vamos a poder ajustar la profundidad de color. Cada punto de la imagen tiene codificado un color con un número determinado de bits.

- Utilizando 8 bits conseguiremos 256 colores
- con 16 bits obtendremos 65.536 colores,
- con 24 bits conseguiremos incluso más colores de los perceptibles con una paleta de 16.777.216 colores distintos. Se llama el color verdadero.



- con 32 bits conseguimos los 16.777.216 colores distintos más 256 niveles de transparencia.

3.2.4. Tipos de formatos

Podemos pensar en una imagen como en un archivo en donde se contiene información sobre cuantos píxeles forma la imagen y para píxel que color se ha de mostrar. El formato más inmediato para guardar ésta información es el **formato RAW** en donde información no se guarda comprimida de ninguna forma y en consecuencia no se pierde ninguna información pero ocupa mucho espacio. Es un formato con una profundidad de unos 36 o 48 bits en donde se guarda la totalidad de datos que ha podido captar el sensor de la cámara digital. En consecuencia no es un formato útil para la web. Los formatos principales en el desarrollo de aplicaciones web son:

- **El formato JPG** (Joint Photographic Experts Group) soporta una profundidad de hasta 24 bits sin transparencias, es el más utilizado en la web y es el más indicado para fotografías. En vez de guardar todos los píxeles de colores distintos va eliminando aquella información que al ojo humano le cuesta más distinguir. Permite distintos grados de compresión, pero contra mayor sea mayor va a ser la información que se eliminará y en consecuencia menor va a ser la calidad de imagen
- **El formato GIF** tiene una paleta limitada a 256 colores con transparencias (una profundidad de 8 bits). Utiliza un sistema de compresión LZW patentado y permite crear animaciones. Dado su limitado número de colores no se suele utilizar mucho en la web excepto para generar imágenes animadas.
- **El formato PNG** (Portable Network Graphics) soporta una profundidad de hasta 24 bits con transparencias. A demás proporciona una compresión de imágenes sin pérdida a diferencia del JPG. Aunque ocupa un poco más que JPG es perfecto para trabajar con transparencias y para imágenes de alta calidad.

3.2.5. Programas para crear y manipular imágenes

Existe una gran cantidad de herramientas dedicadas a la creación y manipulación de imágenes tanto de manera online como aplicaciones instalables. De todas las gratuitas nombradas aquí podréis encontrar un link a su acceso en el apartado de recursos.

Una de las herramientas comerciales más conocidas para la manipulación de imágenes es Adobe Photoshop. Es una herramienta profesional de pago.



Como alternativa a Photoshop en el ámbito de software libre una de la herramienta más conocida es GIMP (GNU Image Manipulation Program). Un proyecto publicado bajo la licencia general de GNU.

También hay muy buenas herramientas para la edición de imágenes online como Photopea o Photoshop Express.

Podemos encontrar otras herramientas especializadas para la optimización de imágenes como el programa gratuito RIOT (Radical Image Optimisation Tool) o la aplicación online Compressor.io.

En el siguiente vídeo vamos a ver las bases para utilizar GIMP.



Uf2_3_2_5_Uso_basicoGimp.wmv: Uso básico de Gimp

3.2.6. **Css icon sprite**

A parte del peso de una imagen, otra característica que va a ralentizar la carga completa de una página web va a ser el número de recursos que vaya a tener que pedir para mostrarla. Cada recurso (archivo CSS, JS, imagen, fuente, etc.) que necesite para cargar la web va a ser una nueva consulta que el navegador va a tener que generar, esperar y resolver. Éste lo podemos menguar si agrupamos el conjunto de iconos que vayamos a utilizar en nuestra web en una sola imagen y aprovechamos CSS para mostrar una parte concreta de la imagen como imagen de fondo.

En el siguiente ejemplo vamos a incluir 6 iconos distintos en 6 DIV. En éste caso los iconos serán círculos de colores de 35x35px más 1 píxel en blanco de margen. En vez de utilizar 6 imágenes distintas hemos creado una sola imagen con los 6 iconos. Así el primer icono se



ha situado en la posición 0,0, el icono rojo en la posición 38,0, el naranja en la posición 75,0, el amarillo en la posición 0,38, el verde en la posición 38,38 y el azul en la posición 75,38. Conociendo su posición dimensionamos cada DIV al tamaño que tienen los iconos y mediante la propiedad “background-position” indicamos que muestre la imagen de fondo pero a partir de la posición que nos interese. Atención, ten presente que el eje de coordenadas se sitúa en la esquina superior izquierda, por lo que las coordenadas del CSS van a ser negativas.

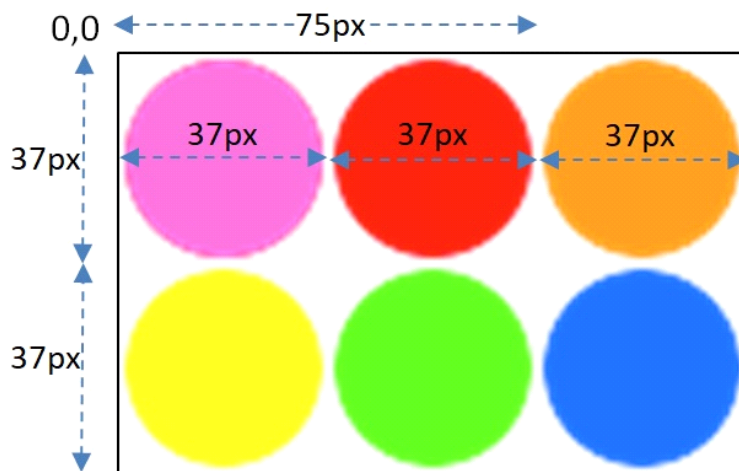


Imagen con los 4 iconos que vamos a utilizar

```
<div class="iconRosa"></div>
<div class="iconRojo"></div>
<div class="iconNaranja"></div>
<div class="iconAmarillo"></div>
<div class="iconVerde"></div>
<div class="iconAzul"></div>
```

```
25
26 .iconRosa,.iconRojo,.iconNaranja,
27 .iconAmarillo, .iconVerde,.iconAzul{
28     background-image: url("icons.png");
29     width: 37px;
30     height: 37px;
31 }
32 .iconRosa{
33     background-position: 0px 0px;
34 }
35 .iconRojo{
36     background-position: -38px 0px;
37 }
38 .iconNaranja{
39     background-position: -75px 0px;
40 }
41 .iconAmarillo{
42     background-position: 0px -38px;
43 }
44 .iconVerde{
45     background-position: -38px -38px;
46 }
47 .iconAzul{
48     background-position: -75px -38px;
49 }
50
```

Código HTML y CSS utilizado para mostrar en cada div uno de los iconos de la imagen



3.3. Audio en la web

Añadir audio en la web es un punto delicado en el diseño web. En general deberemos evitar a toda costa reproducir un audio de forma predeterminada cuando un usuario visite una web. Debemos de pensar que reproducir un audio es una invasión hacia el usuario, ya que es posible que éste esté escuchando otra música o en un espacio de silencio. Por ello el audio debería reproducirse o bien por expresa voluntad del usuario o bien cuando ése sea el principal fin de la página web (por ejemplo, una felicitación navideña).

3.3.1. Formatos

Igual que ocurre con las imágenes hay distintos formatos para guardar en un archivo la información necesaria para reproducir un audio. Cada formato aplicará una compresión distinta a los datos y en consecuencia su calidad y su tamaño serán distintos. Vamos a centrarnos en los tres formatos soportados por la mayoría de navegadores:

- **WAV**: es un formato propiedad de Microsoft e IBM que suele no utilizar ninguna compresión de datos. Ocupa mucho espacio por lo que no suele ser utilizado a no ser que el ruido que se quiera utilizar sea realmente corto.
- **MP3** (MPEG-1 o MPEG-2): un formato desarrollado por Karlheinz Brandenburg que revolucionó la compresión de audio ofreciendo una gran calidad con un tamaño muy pequeño gracias a ir eliminando aquellos sonidos que al oído humano le cuestan más distinguir. Su principal hándicap fue que era una codificación patentada, por lo que cualquier dispositivo que lo quisiera utilizar debería pagar royalties a su autor. A partir del 2017 la pte ha expirado y ahora es de uso libre.
- **OGG**: un formato libre y abierto como alternativa al MP3. Desarrollado y mantenido por la fundación Xiph.org consigue mejor calidad que el MP3 aunque su uso no es tan popular. Ogg realmente permite contener audio, vídeo y texto, y en concreto el formato de codificación del audio es el Vorbis y fue desarrollado por Christopher Montgomery.

Actualmente el formato más ampliamente soportado por los navegadores es MP3 (soportado como mínimo por IE, Chrome, Firefox, Safari, Opera, Vivaldi). El formato OGG no es soportado por Safari ni IE, y el formato WAV no es soportado por IE.



3.3.2. Programas para manipular audios

Actualmente uno de los programas gratuitos y de código abierto más utilizado para la edición de audio es el Audacity.

También es especialmente interesante la suite Free Studio desarrollado por DVDVideoSoft, que contiene un conjunto de programas para editar imágenes, audios y vídeos pero con el que tenemos que ir con cuidado en su instalación para evitar que nos instale plugins en el navegador.

Para convertir archivos de video y audio a distintos formatos podemos utilizar software instalable como Format Factory, o encontrar por internet múltiples opciones online.

3.3.3. Insertar audios

Podemos añadir una música a una página web con la etiqueta <audio>. Ésta etiqueta provocará que el navegador genere un sencillo reproductor de música para poder iniciar, detener, bajar o subir el volumen del audio. Podemos añadir una serie de atributos a la etiqueta <audio> para modificar algunas propiedades del reproductor. La música que queramos reproducir deberá ser accesible desde el servidor.

En concreto los atributos que podremos definir a una etiqueta <audio> son:

- **controls**: indica que se quieren mostrar controles para la reproducción del audio.
- **autoplay**: indica que se quiere reproducir el audio automáticamente una vez se haya cargado.
- **loop**: indica que una vez finalizado el audio se vuelva a reproducir desde el inicio.
- **muted**: especifica que el audio esté silenciado.
- **preload**: sugerimos al navegador cuando queremos que el navegador cargue en memoria el audio. Sus posibles valores son:
 - **none**: el audio no se carga hasta que se quiera reproducir.
 - **auto**: indica que queremos que el audio se cargue cuando se haya cargado la página web.
 - **metadata**: cuando se haya cargado la página web queremos que se carguen los metadata de las canciones.

El elemento <audio> debe contener como mínimo un nodo <source> en el que se indique la ruta al archivo que contiene el audio y el tipo de codificación. Estos dos parámetros se indican con los atributos:

- **src**: ruta relativa desde el HTML actual hasta el documento de audio que se quiera reproducir.
- **type**: indica el códec que ha de permitir interpretar el archivo que se quiere reproducir. Según el formato los posibles valores son:



- **audio/mpeg** : para archivos MP3
- **audio/ogg** : para archivos OGG
- **audio/wav**: para archivos WAV

Como algunos navegadores no soportan todos los formatos de audio, vamos a poder definir distintos `<source>` para vincular distintos archivos con distintos formatos.

Si el navegador no fuera capaz de interpretar la etiqueta `<audio>`, se mostraría el texto incluido dentro del elemento `<audio>`

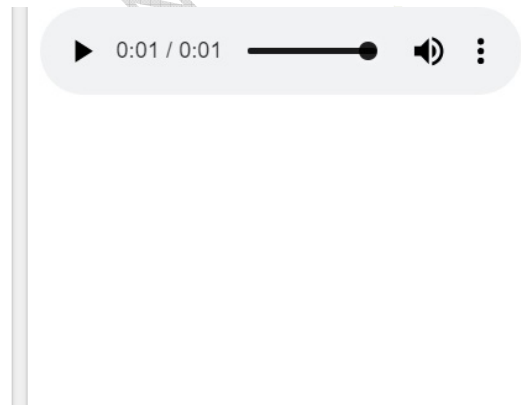
Hay que tener presente que un elemento `<audio>` solo permite reproducir un único archivo de audio, no es una lista de reproducción. Si queremos reproducir distintos audios en un mismo `<audio>` vamos a necesitar modificar el `<source>` con JavaScript.

En el siguiente ejemplo se muestra como cargar una canción de nombre "ejemplo.ogg" ubicada en la misma carpeta que el documento HTML. A demás se especifica que se quieren mostrar los controles para controlar la reproducción y por si el navegador no soporta archivos OGG le indicamos otra versión en formato MP3.

```
<!DOCTYPE html>
<html>
<body>

<audio controls autoplay preload="auto">
  <source src="ejemplo.ogg" type="audio/ogg">
  <source src="ejemplo.mp3" type="audio/mpeg">
  Tu navegador no soporta audio HTML5
</audio>

</body>
</html>
```



Ejemplo de cómo incluir un audio en una página HTML

3.4. Videos en la web

El uso de vídeos es muy parecido al uso de audios en la web. Debido a su peso no se recomienda utilizarlo como elemento decorativo. Deberíamos usar los videos solamente cuando el usuario lo requiera o si forma parte de la funcionalidad de la aplicación web.

Los archivos de video suelen ser contenedores formados por la compresión visual, la compresión del audio o audios y la compresión de otra información como podrían ser los subtítulos. La compresión y descompresión de cada parte se puede llevar a cabo con un códec independiente del resto para maximizar su eficacia.

Cabe destacar que un mismo formato puede utilizar distintos códec dependiendo de la compresión y calidad con la que queramos almacenar la información. Para la web los tres formatos utilizados son:



- **MP4:** soportado por las últimas versiones de todos los navegadores permite reproducir vídeos en formato MPEG-4 con el códec de vídeo H264 y el códec de audio AAC. Su patente no tiene licencia libre por lo que posiblemente algunos navegadores dejen de darle soporte.
- **WebM:** no está soportado por IE ni Safari. Utiliza el códec de vídeo VP8 y el códec de audio Vorbis. Todas sus licencias son gratuitas.
- **Ogg:** igual que WebM no está soportado por IE ni Safari. Utiliza el códec de vídeo Theora y el códec de audio Vorbis. Todas sus licencias son gratuitas.

3.4.1. Insertar videos

Podemos añadir un vídeo a una página web con la etiqueta `<video>`. Ésta etiqueta provocará que el navegador genere un sencillo reproductor de vídeo para poder iniciar, detener, bajar o subir el volumen del vídeo. Podemos añadir una serie de atributos a la etiqueta `<video>` para modificar algunas propiedades del reproductor. El vídeo que queramos reproducir deberá ser accesible desde el servidor.

En concreto los atributos que podremos definir a una etiqueta `<video>` son:

- **controls:** indica que se quieren mostrar controles para la reproducción del vídeo.
- **autoplay:** indica que se quiere reproducir el vídeo automáticamente una vez se haya cargado.
- **loop:** indica que una vez finalizado el vídeo se vuelva a reproducir desde el inicio.
- **muted:** especifica que el vídeo esté silenciado.
- **height:** especifica la altura del reproductor de vídeo.
- **width:** especifica la anchura del reproductor de vídeo.
- **poster:** especifica la ruta hacia una imagen para ser mostrada hasta que empiece la reproducción del vídeo
- **preload:** sugerimos al navegador cuando queremos que el navegador cargue en memoria el audio. Sus posibles valores son:
 - **none:** el audio no se carga hasta que se quiera reproducir.
 - **auto:** indica que queremos que el audio se cargue cuando se haya cargado la página web.
 - **metadata:** cuando se haya cargado la página web queremos que se carguen los metadatos de las canciones.

El elemento `<video>` debe contener como mínimo un nodo `<source>` en el que se indique la ruta al archivo que contiene el vídeo y el tipo de codificación. Estos dos parámetros se indican con los atributos:



- **src**: ruta relativa desde el HTML actual hasta el documento de video que se quiera reproducir.
- **type**: indica el códec que ha de permitir interpretar el archivo que se quiere reproducir. Según el formato los posibles valores son:
 - **video/mp4** : para archivos MP4
 - **video/webm** : para archivos WebM
 - **video/ogg**: para archivos Ogg

Como algunos navegadores no soportan todos los formatos de video, vamos a poder definir distintos `<source>` para vincular distintos archivos con distintos formatos.

Si el navegador no fuera capaz de interpretar la etiqueta `<video>`, se mostraría el texto incluido dentro del elemento `<video>`

Hay que tener presente que un elemento `<video>` solo permite reproducir un único archivo de video, no es una lista de reproducción. Si queremos reproducir distintos videos en un mismo `<video>` vamos a necesitar modificar el `<source>` con JavaScript.

En el siguiente ejemplo se muestra como cargar un video de nombre "mi_video.mp4" ubicada en la misma carpeta que el documento HTML. A demás se especifica que se quieren mostrar los controles para controlar la reproducción y por si el navegador no soporta archivos MP4 le indicamos otra versión en formato OGG.

```
<body>

  <video width="320" height="240"
    poster="poster_video.jpg" controls>
    <source src="mi_video.mp4" type="video/mp4">
    <source src="mi_video.ogg" type="video/ogg">
    Tu navegador no soporta videos HTML5
  </video>

</body>

</html>
```



3.5. Animar con CSS

Con CSS3 vamos a poder alterar propiedades de nuestros elementos dando un efecto de animaciones. Existen dos tipos de animaciones posibles con CSS: las transiciones y las animaciones.



3.5.1. Transiciones

Por defecto cuando cambiamos la propiedad de un elemento HTML éste cambio se produce de forma inmediata. Por ejemplo, en el siguiente código hemos indicado que al situar el ratón encima del elemento *.transicion* cambie su color de fondo y sus tamaños del elemento. Estos cambios los realizará en un solo paso:

```
<div class="transicion">  
  EJEMPLO  
</div>  
  
</body>  
  
</html>
```

```
51 .transicion{  
52   width: 75px;  
53   height: 50px;  
54   background-color: #FF0000;  
55 }  
56 .transicion:hover{  
57   width: 150px;  
58   height: 100px;  
59   background-color: #00FF00;  
60 }  
61 }
```



Ejemplo de cambio de propiedades sin transición

Añadiendo la propiedad “transition” vamos a poder especificar que cuando se produzca un cambio en el valor de alguna propiedad, efectúe este cambio con una suave transición. En el siguiente ejemplo indicamos que cuando se cambie cualquier propiedad del elemento *.transicion* se realice el cambio mediante una animación de 3 segundos de tipo “linear”.

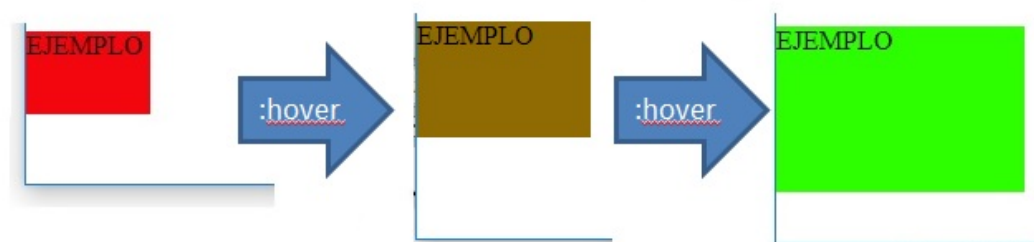


```
<div class="transicion">
  EJEMPLO
</div>

</body>

</html>
```

```
51 .transicion{
52     width: 75px;
53     height: 50px;
54     background-color: #FF0000;
55     transition-property: all;
56     transition-duration: 3s;
57     transition-timing-function: linear;
58 }
59 .transicion:hover{
60     width: 150px;
61     height: 100px;
62     background-color: #00FF00;
63 }
64 }
```



Ejemplo de cambio de propiedades definiendo una *transition*

Los distintos parámetros que podemos especificar para una transición son:

- **transition-property:** (por defecto all) le asignaremos como valor aquella propiedad CSS a la que queramos aplicarle una transición cuando su valor varíe. Si queremos aplicar una transición cuando varíe cualquier propiedad CSS indicaremos el valor "all". Por ejemplo: *transition-property: width;* solo aplicaría la transición definida cuando se modificara la propiedad *width*.
- **transition-duration:** (0 por defecto) le asignamos en segundos (s) o milisegundos (ms) la duración de una animación.
- **transition-timing-function:** especifica la curva de velocidad de la animación. Sus principales valores posibles son:
 - linear: la animación irá igual de rápido hasta que finalice.
 - ease: (valor por defecto) la animación empieza de forma lenta, luego acelera y finalmente se ralentiza de nuevo hasta finalizar.
- **transition-delay:** (por defecto 0) le asignamos en segundos (s) o milisegundos (ms) la cuanto tiempo queremos que tarde en empezar una animación.



3.5.2. Animaciones

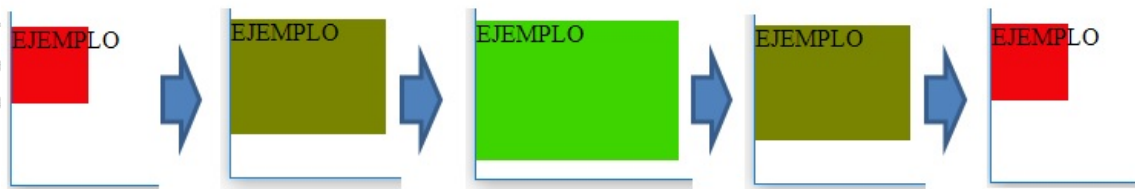
En CSS podemos crear animaciones personalizadas indicando en una escala del 0 al 100% tantos estados de las propiedades como queramos. Estos estados los definiremos en un conjunto de *@keyframes* y les asignaremos un nombre. Después podremos asignar como animación a cualquier regla CSS el nombre del *@keyframe*.

A diferencia de la propiedad *transition*, éstas animaciones se pueden iniciar directamente sin necesidad de que se produzca un cambio en las propiedades, pues el propio *@keyframe* va a indicar los cambios en las propiedades que se deben de producir durante el transcurso de la animación.

En el siguiente ejemplo hemos definido un bloque de *@keyframes* llamado "animateTransition" con 3 estados distintos. En el primer estado el elemento que utilice éste *keyframe* como animación tendrá una anchura y altura de 50px y un color de fondo rojo. La animación se llevará a cabo hasta tener una anchura de 150px, una altura de 50px y un color de fondo verde. Como último paso de la animación se modificará la altura y anchura de nuevo a 50px y establecerá un color de fondo rojo. En éste ejemplo se ha aplicado éste *keyframe* al elemento *.transicion* con una animación de 3 segundos de tipo linear que se volverá a ejecutar una vez se ha finalizado.

```
40 <div class="transicion">
41 |   EJEMPLO
42 </div>
43
44 </body>
45
46 </html>

49 .transicion{
50   width: 75px;
51   height: 50px;
52   background-color: #FF0000;
53   animation: animateTransition 3s linear infinite;
54 }
55 @keyframes animateTransition{
56   0%{ width: 50px;
57       height: 50px;
58       background-color: #FF0000;}
59
60   50%{ width: 150px;
61        height: 100px;
62        background-color: #00FF00;}
63
64   100%{width: 50px;
65        height: 50px;
66        background-color: #FF0000;}
67 }
```



Ejemplo de una animación estableciendo los cambios de las propiedades con *@keyframe*



Los principales parámetros que podemos especificar en una animación son:

- **animation-name:** el nombre del keyframe que contiene los cambios de propiedades que se van a producir.
- **animation-duration:** (0 por defecto) le asignamos en segundos (s) o milisegundos (ms) la duración de una animación.
- **animation-timing-function:** especifica la curva de velocidad de la animación. Sus principales valores posibles son:
 - linear: la animación irá igual de rápido hasta que finalice.
 - ease: (valor por defecto) la animación empieza de forma lenta, luego acelera y finalmente se ralentiza de nuevo hasta finalizar.
- **animation-delay:** (por defecto 0) le asignamos en segundos (s) o milisegundos (ms) la cuanto tiempo queremos que tarde en empezar una animación.
- **animation-iteration-count:** (por defecto 1) podemos indicar el número de veces que se repita una animación. Como valor especial podemos indicar "infinite" para indicar que una animación se debe repetir infinitas veces.
- **animation-direction:** (por defecto "normal") podemos indicar si una animación debe seguir los @keyframes de 0% 100% o de 100% a 0%. El valor "alternate" realiza la animación desde el final hasta el inicio.



Test de autoevaluación

¿Qué formato de audio tiene una menor compresión?

- a) WAV
- b) MP3
- c) Mpeg1
- d) OGG

¿Qué etiqueta HTML permite insertar un audio en un HTML?

- a) media
- b) sound
- c) include
- d) audio

¿Qué indica la propiedad CSS “transition”?

- a) queremos alterar una propiedad CSS con una transición.
- b) cuando se produzca un cambio en una propiedad, que se aplique una transición.
- c) queremos cambiar una etiqueta HTML por otra con una transición.
- d) queremos animar una etiqueta HTML has que sus propiedades tengan unos ciertos valores.



Recursos y enlaces

- Sitio web de Creative Commons: <https://creativecommons.org>
- Gimp, editor de imágenes gratuito <https://www.gimp.org>
- Photopea, editor de imágenes online y gratuito: <https://www.photopea.com>
- Photoshop Express, editor de imágenes online y gratuito: <https://www.photoshop.com/products/photoshopexpress>
- Riot, optimizador de imágenes: <http://luci.criosweb.ro/riot/>
- Kraken, optimizador online de imágenes: <https://kraken.io/web-interface>
- Compressor.io, optimizador online de imágenes: <https://compressor.io>
- Audacity, editor de audio <https://sourceforge.net/projects/audacity/>
- Format Factory, conversor de audio y video a distintos formatos: <http://www.pcfreetime.com/>
- Free Studio: suite de software para la edición de audio, video e imágenes: <https://www.dvdvideosoft.com/es/free-dvd-video-software-download.htm>
- Aplicación web para crear un logo online: <https://www.crearlogogratisonline.com>
- Banco de audios copyleft: <http://freemusicarchive.org/about>
- Banco de audios CC BY SA: <http://opsound.org/info/license/>
- Buscador de contenidos Creative Commons: <https://search.creativecommons.org>
- Buscador de contenidos Creative Commons: <https://ccsearch.creativecommons.org>

Conceptos clave

- **Elemento:** un elemento html es el conjunto de una etiqueta inicial, su contenido y la etiqueta final.
- **Css icon sprite:** imagen que contiene distintos iconos. Mediante CSS insertaremos como imagen de fondo uno de los iconos contenidos.
- **Licencias Creative Commons (CC)** permiten a los autores ceder bajo ciertas condiciones algunos de los derechos sobre sus obras.
- **96ppp** es la resolución aconsejada para las imágenes en la mayoría de páginas web.



Ponlo en práctica

Actividad 1

Crea con un editor gráfico como GIMP una imagen PNG con transparencias formada por 3 distintos iconos de 45x45 px cada uno. Utiliza la técnica del “icon sprites” para mostrar cada uno de los tres iconos como fondo de un botón.

Añade además un DIV de color inicial rojo que con una transición infina (en CSS) se mueve en diagonal hacia abajo y hacia arriba cambiando su color de rojo a verde.



SOLUCIÓN



Tema 4: Integración de contenido interactivo

¿Qué aprenderás?

- Reconocer los principios básicos de la comunicación visual.
- Crear hojas de estilos seleccionando los colores y tipografías adecuadas a la aplicación.
- Desarrollar wireframes para el diseño de interfaces.
- Maquetar la web siguiendo los wireframes y las hojas de estilos

¿Sabías que...?

- jQuery fue lanzada oficialmente el 26 de agosto de 2006
- Hay unos 61.382.162 sitios web activos que utilizan jQuery.
- Microsoft y Nokia anunciaron que incluirán jQuery en sus plataformas.
- La principal ventaja que ofrecía jQuery cuando nació fue la compatibilidad con todos los navegadores, algo impensable en la época.
- jQuery está mantenido y actualizado por la JS Foundation. Son miembros de esta asociación IBM, Samsung, Microsoft, Google.



4. Web interactiva

4.1. Introducción a jQuery

Aunque con CSS es posible realizar muchos tipos distintos de animaciones, en muchas ocasiones necesitaremos configurar interacciones del usuario junto con animaciones más complejas. Necesitaremos entonces alterar los estilos de elementos cuando se produzcan ciertos eventos mediante JavaScript.

jQuery es una popular librería (conjunto de funciones) programada con JavaScript con el objetivo de facilitar las tareas más comunes en JS. Ésta librería está concentrada en un único documento JS que vamos a vincular a nuestro HTML para poder utilizar sus funciones. Nunca vamos a modificar el archivo JS que contiene la librería, sino que crearemos otro archivo JS en el que escribiremos nuestro código.

4.1.1. Vincular la librería jQuery

Tenemos dos caminos para vincular jQuery a la web:

- La primera opción es utilizar un repositorio online (un CDN) para obtener el archivo. Éste método permite ahorrar ancho de banda al servidor de nuestro sitio web debido a que la librería se va encontrar en el servidor del repositorio y no en el nuestro. Otra ventaja es que si otro sitio web utiliza también el mismo archivo del mismo repositorio, el navegador posiblemente no vuelva a descargar el archivo y utilice su versión almacenada en memoria disminuyendo la velocidad de carga de la web. Su principal desventaja es el poco control que tenemos sobre el archivo y que no lo podremos utilizar cuando estemos desarrollando sin internet. Por eso éste método se recomienda utilizar cuando subamos la página al servidor.

Hay distintos CDN libres para utilizar. Google nos ofrece el suyo en: <https://developers.google.com/speed/libraries>

- La segunda opción es descargar la librería de la página web oficial y vincular el archivo JS con el HTML. Éste método nos ofrece un control absoluto sobre la librería. Como la tendremos descargada en local, cuando desarrollemos el sitio web la podremos utilizar aún sin estar conectados. Por el contrario cada vez que un usuario distinto pida el documento nuestro sitio web gastará ancho de banda en enviarlo.

Nos podemos descargar la librería desde la web oficial: <https://jquery.com>

En ambos casos es muy importante vincular la librería jQuery antes de los archivos JS que vayamos a programar para que las funciones definidas en jQuery estén disponibles para nuestras funciones JS.



En el siguiente ejemplo vinculamos la librería jQuery desde el CDN de google y un archivo JS de nombre miArchivo.JS:

```
<head>
  <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>

  <script type="text/javascript" src="miArchivoJS.js"> </script>

</head>
```

En el siguiente ejemplo vinculamos la librería jQuery descargada y un archivo JS de nombre miArchivo.JS:

```
<head>
  <script type="text/javascript" src="jquery-1.9.1.min.js"> </script>

  <script type="text/javascript" src="miArchivoJS.js"> </script>

</head>
```

4.1.2. Empezando con jQuery

Igual que ocurre con JavaScript cuando queramos acceder a un elemento del HTML para modificar sus propiedades o vincularle un evento vamos a tener que estar seguros que el HTML se ha cargado completamente. Por eso siempre que queramos utilizar jQuery en nuestra aplicación web vamos a tener que declarar en nuestro JS una función inicial y llamarla cuando se el documento HTML esté listo mediante la sintaxis `$(document).ready(función_inicial);`. En ésta función inicial escribiremos todo el código que queramos ejecutar cuando se haya cargado la web.

El siguiente ejemplo se ha declarado una función de nombre “iniciar” que se utilizará como función inicial una vez cargada la web:

```
$(document).ready(iniciar);

function iniciar(){
    alert("hola");
}
```



También es posible definir la función inicial como una función anónima:

```
$(document).ready( function (){ alert("hola"); } );
```

4.1.3. Mecánica de jQuery

La mecánica de jQuery se basa en dos pasos simples:

- e) primero selecciona un elemento HTML
- f) y luego indica qué hacer con él.

Para seleccionar un elemento HTML se indica como parámetro de la función `$()` el selector CSS que lo seleccionaría. De ésta forma podemos seleccionar múltiples elementos. El siguiente ejemplo selecciona el elemento con id `#miDiv`, todos los `SPAN` y los `LI` impares: Una vez seleccionados los elementos sobre los que queremos operar se llaman a las funciones jQuery para indicar qué hacer con ellos. En jQuery es común que una función sea del tipo `set` o `get` dependiendo del número de parámetros indicados. Vamos a empezar por ver cómo podemos modificar estilos CSS con jQuery

```
$("#miDiv , span, li:nth-child(2n+1)")
```

4.2. Modificar estilos con jQuery

Para modificar las propiedades CSS u obtener sus valores vamos a utilizar la función `.css`. Podemos modificar una propiedad CSS con la sintaxis: `.css('nombre-propiedad','valor');`. El siguiente ejemplo modifica el color de fondo de todos los `div`:

```
$('div').css('background-color','#FF55FF');
```

Para obtener el valor de una propiedad CSS utilizaremos igualmente la función `.css` pero pasando solo como parámetro el nombre de la propiedad de la que queremos obtener su valor. El siguiente ejemplo obtiene el color de la letra de `#miDiv`

```
var color = $('#miDiv').css('color');
```

jQuery nos ofrece una forma muy sencilla de aumentar o reducir un valor numérico de una propiedad CSS. Solo vamos a tener que indicar como valor de la propiedad que queremos modificar el operador `+=` o bien `-=` seguido de la cantidad que queramos incrementar o disminuir. El siguiente ejemplo aumenta en 20px el padding de todos los elementos de la clase `.miSpan`:



```
$('.miSpan').css('padding','+=20');
```

A demás jQuery nos ofrece la posibilidad de indicar más de una propiedad CSS pasando como único parámetros una notación JSON con el conjunto de propiedades y valores que queremos establecer. En el siguiente ejemplo utilizamos la notación JSON para aumentar en 20px el padding y cambiar el color de fondo de todos los elementos de la clase .miSpan:

```
$('.miSpan').css({'padding':'+=20', 'background-color':'#FF55FF'});
```

jQuery define también un conjunto de funciones específicas para ocultar y mostrar elementos. Las principales son:

- **.hide()** : oculta los elementos seleccionados.
- **.show()**: muestra los elementos seleccionados.
- **.toggle()**: si los elementos seleccionados están visibles los oculta y viceversa.
- **.fadeOut()**: oculta los elementos seleccionados con un difuminado.
- **.fadeIn()**:muestra los elementos seleccionados con un difuminado.
- **fadeToggle()**:si los elementos seleccionados están visibles los oculta y viceversa con un difuminado.

Hasta ahora hemos visto como modificar estilos CSS pero no hemos visto ninguna interacción. Vamos a ver ahora cómo podemos vincular los distintos eventos que puede lanzar el usuario.

4.3. Eventos con jQuery

Capturar un evento con jQuery e muy sencillo, solo tendremos que:

- a) primero seleccionar el elemento sobre el que queremos capturar el evento. Por ejemplo: `$('#miDiv')`
- b) seguidamente indicaremos la función jQuery que capture el evento que queramos y pasaremos como parámetro la función que querremos ejecutar cuando se lance el evento. Por ejemplo, para que al clicar sobre sobre #miDiv se llame a la función "miFuncion" escribiremos: `$('#miDiv').click(miFuncion)`

El siguiente ejemplo muestra un código completo que vincula un evento que al clicar muestre un alert:



```
$(document).ready(iniciar);  
function iniciar(){  
    $('#miDiv').click(miFuncion);  
}  
function miFuncion(){  
    alert('has clicado encima un #miDiv');  
}
```

En la documentación oficial de jQuery disponible en la misma página web podemos encontrar el listado de todos los eventos. Los principales tipos de eventos que podemos capturar en jQuery son:

- **.click()** : se ejecuta al clicar encima de los elementos seleccionados.
- **.dblclick()** : se ejecuta al hacer dobleclick encima de los elementos seleccionados.
- **.mouseenter()**: se ejecuta cuando el ratón entra dentro de los elementos seleccionados.
- **.mouseover()** : se ejecuta cuando el ratón está situado encima los elementos seleccionados.
- **.mouseleave()**: se ejecuta cuando el ratón sale de los elementos seleccionados.
- **.hover()** : se ejecuta cuando el ratón entra o sale de los elementos seleccionados.

Podemos desvincular un evento de un elemento con la función **.off()** indicando como parámetro el nombre del evento que queremos desvincular. Si no indicamos ningún nombre se desvincularan todos los eventos de los elementos seleccionados. En el siguiente ejemplo primero le vinculamos un evento *click* y *dblclick* pero luego le desvinculamos el evento *dblclick*:

```
$('#miDiv').click(miFuncion);  
$('#miDiv').dblclick (miFuncion2);  
  
$('#miDiv').off("dblclick");
```

Cuando se ejecute una función mediante un evento vamos a poder obtener el elemento que ha lanzado la función mediante el selector **\$(this)**. En el siguiente ejemplo utilizando **\$(this)** para modificar el color de fondo del DIV clicado a azul:



```
iplo.html x
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>

<script type="text/javascript" src="ejemplo.js"></script>
</head>
<body>

<div> DIV1 </div>
<div> DIV2 </div>
<div> DIV3 </div>

JS ejemplo.js x
1 $(document).ready(iniciar);
2 function iniciar(){
3     $('div').click(changeColorBlue);
4
5
6 }
7 function changeColorBlue(){
8     $(this).css({"background-color":"blue"});
9 }
10
11
```

4.4. Animaciones con jQuery

jQuery contiene un conjunto de animaciones predefinidas pero también nos ofrece la posibilidad de crear nuestras propias animaciones fácilmente. En las animaciones indicaremos siempre las propiedades CSS que queremos animar de un conjunto de elementos para que lleguen a tener un cierto valor.

La función **animate()** nos permite crear una animación personalizada sobre los elementos seleccionados indicando como primer parámetro un JSON con las propiedades CSS que queremos modificar y como segundo parámetro y un JSON con las propiedades de la animación. El siguiente ejemplo anima a todos los elementos de la clase `.block` para que durante un segundo se sitúen a 100px a la izquierda:

```
$(".block ").animate(
    { left: "100px" },
    { duration: 1000 }
);
```

Solo se pueden crear animaciones para propiedades con valores numéricos. Las propiedades CSS no numéricas no se tendrán en cuenta.

Algunas de las principales propiedades de la animación que podemos configurar indicándolas en un JSON como segundo parámetro de la animación son:

- **duration:** le podemos indicar con un valor numérico el número de milisegundos que queremos que dure la animación.
- **queue:** admite o bien *false* o bien *true*. En el caso que a un elemento se le asigne más de una animación por defecto solo se empezará con la segunda animación una vez se haya terminado la primera. Si ésta propiedad tiene un valor "false" se empezará la animación inmediatamente.
- **progress:** le podemos indicar la función que queremos que se ejecute en cada paso de la animación.



- **complete** le podemos indicar la función que queremos que se ejecute cuando se haya terminado la animación.
- **start:** le podemos indicar una función para que se ejecute en el momento que empiece la animación.

En el siguiente ejemplo inicialmente utilizamos jQuery para establecer las propiedades CSS del elemento `#divAnimado` y aplicar una animación que incremente su anchura en 50 píxeles, establezca su posición superior a 100px durante 3 segundos. A demás se ha configurado al animación porque en cada paso de la misma se llame a la función `muestraAchura()` mostrando dentro suyo su anchura actual y que cuando se haya completado la animación se llame a `muestraFinal()` para mostrar el mensaje "FIN ANIMACION".

VERSIÓN IMPRIMIBLE ALUMNO LINK



lo.html x

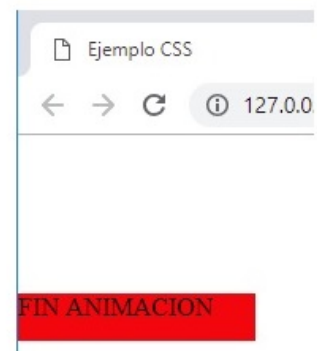
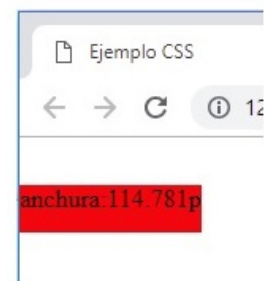
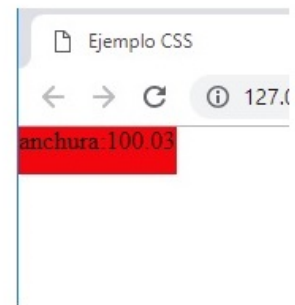
```
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>

<script type="text/javascript" src="ejemplo.js"></script>
</head>
<body>
  <div id="divAnimado">

  </div>
```

JS ejemplo.js

```
1 $(document).ready(iniciar);
2 function iniciar() {
3   $("#divAnimado").css({
4     "height": "30px",
5     width: "100px",
6     left: "0",
7     top: "0",
8     position: "absolute",
9     "background-color": "red"
10  });
11  iniciaAnimacion();
12 }
13
14 function iniciaAnimacion() {
15   $("#divAnimado").animate(
16     {
17       width: "+=50",
18       top: 100
19     }, {
20     duration: 3000,
21     progress: muestraAnchura,
22     complete: muestraFinal
23   });
24 }
25
26
27
28 function muestraAnchura(){
29   let anchura = $(this).css("width");
30   $(this).html("anchura: "+anchura);
31 }
32 function muestraFinal(){
33   $(this).html("FIN ANIMACION");
34 }
```



Ejemplo de utilización de jQuery para modificar los estilos, el contenido y aplicar una animación a un elemento



4.5. jQuery DOM

jQuery nos ofrece distintas funciones para modificar el DOM creando nuevos elementos y modificando sus atributos y contenido.

Vamos a empezar viendo aquellas funciones que nos permiten modificar las propiedades de los elementos seleccionados:

- **.html()** : obtiene el contenido de los elementos seleccionados.
- **.html("nuevo contenido")** : modifica el contenido de los elementos seleccionados con el texto introducido. Si se introduce un código HTML es interpretado por el navegador.
- **.text()** : retorna todos los textos que contenga los elementos seleccionados. Si un elemento seleccionado contiene otros, retornará también el contenido de sus descendientes.
- **.text('<p>text</p>')** : modifica el contenido de los elementos seleccionados con el texto introducido. Si se introduce código HTML se insertarán los caracteres necesarios para que el navegador no lo interprete como HTML y muestre ese mensaje por pantalla. En el ejemplo el contenido será: '<p>text</p>' .
- **.attr("id")**: obtiene el valor del atributo *id* de los elementos seleccionados.
- **.attr("id","nuevoValor")**: establece el valor del atributo *id* de los elementos seleccionados con el nuevo valor indicado.
- **.addClass("rojo")**: añade todos los elementos seleccionados la clase "rojo".
- **.toggleClass("rojo")**: quita o añade la clase 'rojo' a todos los elementos seleccionados.
- **.hasClass("rojo")**: retorna true si algún elemento es de la clase "rojo".

jQuery también permite crear y eliminar elementos. Vamos a ver las funciones principales para crear elementos:

- **let nuevoElem= \$("<div> nuevo DIV </div>")** : mediante el selector **\$()** podemos crear un nuevo elemento con el código HTML incluido dentro y guardar este elemento en una variable. Estos elementos no son compatibles con el tratamiento del DOM que hace JS. El ejemplo crea un tag DIV con el text "nuevo DIV"
- **.append(nuevoElem)** : añade el elemento guardado en la variable *nuevoElem* como último elemento dentro de todos los elementos seleccionados.
- **.prepend(nuevoElem)** : añade el elemento guardado en la variable *nuevoElem* como primer elemento dentro de todos los elementos seleccionados.
- **.remove(nuevoElem)** : añade el elemento guardado en la variable *nuevoElem* como último elemento dentro de todos los elementos seleccionados.



- `$("#elem").before(nuevoElem)` :añade el elemento guardado en la variable *nuevoElem* delante del elemento con id *elem*.
- `$("#elem").before(nuevoElem)` :añade el elemento guardado en la variable *nuevoElem* delante del elemento con id *elem*.
- `nuevoElem.remove()`: elimina el elemento guardado en la variable *nuevoElem*.
- `$("#padre").remove("div")`: elimina todos los DIV dentro del elemento con id *padre*.

4.6. Plugins jQuery

Los plugins son pequeñas aplicaciones ya programadas que podemos insertar y utilizar en nuestro sitio web. Una de las principales ventajas de jQuery es la gran cantidad de plugins que existen programados con esta librería. Vamos a ver como ejemplo el plugin slick que nos va a permitir crear fácilmente un conjunto de imágenes que van a ir pasando como un slider o un carrousel.

Podemos descargar el plugin desde la página oficial <http://kenwheeler.github.io/slick/>. No descargará un archivo comprimido con distintos ejemplos para su utilización y una carpeta de nombre "slick" que deberemos copiar en nuestro sitio web y que contendrá los archivos "slick.js", "slick.css" y "slick-theme.css" que deberemos vincular a nuestros HTML después de importarla librería jQuery y antes de nuestro JS.

El plugin se encargará de transformar un DIV como contenedor de distintos elementos en un slider en que los distintos elementos se vayan moviendo. Nosotros deberemos escribir el código JS para indicar qué DIV va a ser el contenedor de los elementos del slider y configurarlo.

Para indicar qué elemento será el contenedor lo seleccionamos y le aplicamos la función slick() pasándole como parámetro y en notación JSON la configuración del plugin.

Los principales parámetros que podemos configurar en slick son:

- **slidesToShow**: acepta un número para indicar el número de diapositivas que se van a mostrar a la vez.
- **slidesToScroll**: acepta un número para indicar el número de diapositivas que se moverán cada vez que se mueva el slider.
- **nextArrow**: permite definir un código HTML que se utilizará como botón para mostrar las siguientes diapositivas.
- **prevArrow**: permite definir un código HTML que se utilizará como botón para mostrar las diapositivas anteriores.



- **dot:** si el valor es "true" muestra unos puntos para seleccionar el número de diapositiva que queremos ver.
- **infinte:** si el valor es "true" después de mostrarse el último elemento se mostrará el primero y viceversa.
- **autoplay:** si el valor es "true" el slider se moverá automáticamente.
- **fade:** si el valor es "true" la transición entre diapositivas se hará mediante un difuminado.
- **responsive:** acepta como parámetro un array con un conjunto de configuraciones JSON para cambiar los parámetros del slider según la anchura de la ventana.

En el siguiente ejemplo creamos un slider utilizando como contenedor el DIV con id "slider" formado por 3 elementos. Una vez cargada la web indicamos con jQuery que #slider será el contenedor y lo configuramos para que el slider muestre dos elementos a la vez, la transición sea de uno en uno con un tiempo de 500 milisegundos en un bucle infinito y poder girar el slider o bien mediante unos puntos o clicando en los textos PREV y NEXT:

```
<> ejemplo.html • JS ejemplo.js x
10 <link href="slick/slick-theme.css" rel="stylesheet" type="text/css" />
11 <link href="slick/slick.css" rel="stylesheet" type="text/css" />
12
13 <script type="text/javascript"
14 src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
15 </script>
16 <script type="text/javascript" src="slick/slick.js"></script>
17 <script type="text/javascript" src="ejemplo.js"></script>
18
19 </head>
20
21
22 <body>
23 <div id="slider">
24 <div> CONTENIDO DEL 1r SLIDER </div>
25 <div> CONTENIDO DEL 2o SLIDER </div>
26 <div> CONTENIDO DEL 3r SLIDER </div>
27
28 </div>
29
```

```
1 $(document).ready(iniciar);
2 function iniciar() {
3     $("#slider").slick({
4         slidesToShow: 2,
5         slidesToScroll: 1,
6         dots: true,
7         infinite: true,
8         speed: 500,
9         prevArrow: '<h2>PREV</h2>',
10        nextArrow: '<h2>NEXT</h2>'
11    });
12 }
```





Uf2_3_2_5_Uso_plugin_slick.wmv : Ejemplo Uso plugin slick

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP



Test de autoevaluación

¿Cuál es el código correcto de jQuery para establecer el color de fondo de todos los elementos div a rojo?

- c) `$("header").css("backgroundColor", "red");`
- d) `$(".div").style("background-color", "red");`
- e) `$("#div").css({"background-color"}, {"red"});`
- f) `$("div").css("background-color", "red");`

¿Qué código jQuery permite añadir el elemento con id "myDiv" dentro de los elementos de clase "myClass"?

- g) `$(".myClass").moveTo("#myDiv");`
- h) `$(".myClass").insert("#myDiv");`
- i) `$("#myDiv").moveTo(".myClass");`
- j) `$(".myClass").append("#myDiv");`

¿Cómo puedo crear un div que cuando lo clique me ejecute "mifuncion"?

- k) `var d = $("<div onclick='mifuncion'>clik</div>");`
- l) `var d = $("<div>clik</div>"); d.click(mifuncion);`
- m) `var d = $("DIV"); d.onclick=mifuncion();`
- n) `var d = $("<div>clik</div>"); d.onclick(mifuncion());`



Recursos y enlaces

- CDN de Google: <https://developers.google.com/speed/libraries>
- jQuery: <https://jquery.com>
- Slick: plugin para jQuery <http://kenwheeler.github.io/slick/>
- <http://youmightnotneedjquery.com> es una web que ofrece alternativas JavaScript a tareas usualmente asignadas a jQuery

Conceptos clave

- **Elemento:** un elemento html es el conjunto de una etiqueta inicial, su contenido y la etiqueta final.
- **DOM:** Document Object Model, es el conjunto de objetos que los navegadores generan a partir de cada elemento html. Cada elemento se corresponde con un objeto.
- **JS:** lenguaje de programación JavaScript.



Ponlo en práctica

Actividad 1

En un documento HTML añade un botón que al clicarlo empiece una animación de 2 segundos para aumentar el tamaño del botón en 50px. En cada segundo se ha de variar aleatoriamente el color del botón. Si el ratón sale del botón hay que detener la animación.



SOLUCIÓN

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP