

Elements multimèdia al web: creació i integració

Xavier Garcia Rodríguez

Disseny d'interfícies

Índex

Introducció	5
Resultats d'aprenentatge	7
1 Elements multimèdia al web: imatges, vídeo i àudio	9
1.1 Elements d'imatge per al web	9
1.1.1 Tipus d'imatges al web	10
1.1.2 Formats d'imatges	16
1.1.3 Programari per crear i processar imatges	21
1.1.4 Optimització d'imatges per a la web	28
1.2 Elements d'àudio i vídeo per al web	34
1.2.1 Programari per manipular i editar àudio i vídeo	34
1.2.2 Àudio: formats i conversions de formats (exportar i importar)	41
1.2.3 Vídeo: codificació de vídeo, conversió de formats (exportar i importar)	52
1.2.4 Integració d'àudio i vídeo en una animació	54
1.2.5 Plataformes de vídeo en 'streaming'	56
1.2.6 Animacions d'imatges i text	61
2 Continguts interactius al web. Introducció a jQuery	85
2.1 Elements interactius bàsics. Introducció a JQuery	85
2.1.1 Configuració dels navegadors per a la visualització d'elements interactius	85
2.1.2 Eines per a la inclusió de contingut multimèdia	91
2.1.3 Introducció a jQuery	92
2.1.4 Elements interactius avançats: biblioteques i connectors	109
2.1.5 Desenvolupament de connectors per a jQuery	112
2.1.6 Introducció a JSON	115
2.1.7 Execució i verificació	119
2.2 Casos pràctics d'integració de continguts multimèdia	122
2.2.1 Creació d'un reproductor de vídeo	123
2.2.2 Creació d'un bàner amb carrega dinàmica de dades	148
2.3 Els drets d'autor i l'ús de les llicències	152
2.3.1 Drets de la propietat intel·lectual	153
2.3.2 Llicències	153
2.3.3 Recursos amb llicències fàcilment localitzables	156

Introducció

Encara que inicialment la web només permetia mostrar text i taules, ara per ara és impensable trobar cap lloc o aplicació web que no compti, com a mínim, amb imatges i algun tipus d'animació (per exemple: efectes als botons, anuncis rotatius, etc.). Això obliga els desenvolupadors a conèixer com treballar-hi i com incloure'ls als seus treballs.

En aquesta unitat es tractaran els tipus de recursos que poden afegir-se per enriquir els llocs i les aplicacions web: tipus d'imatges, àudio, vídeo i animacions. En finalitzar-la sereu capaços de crear els vostres propis recursos, optimitzar elements multimèdia i afegir-los tant a llocs com a aplicacions web.

A l'apartat “Elements multimèdia al web: imatges, vídeo i àudio” s'estudien els tipus d'imatges, les particularitats de cada format i el programari utilitzat per crear-les, editar-les i optimitzar-les, així com l'ús de l'àudio i vídeo al web, quins formats cal utilitzar, com editar-los i el programari que permet fer-ho. També s'aborda la creació d'imatges animades i altres animacions programades amb CSS.

Atesa la importància de la utilització de biblioteques per manipular aquests recursos, l'apartat “Continguts interactius al web. Introducció a jQuery” explica com fer servir la més popular, jQuery; com afegir connectors de tercers i com crear-ne de propis. També s'hi fa un petit repàs de les estructures de dades i de les eines de comprovació.

Com es pot apreciar, aquesta unitat és eminentment pràctica, i per obtenir el màxim aprofitament és imprescindible seguir els exemples que accompanyen les explicacions.

Resultats d'aprenentatge

En finalitzar aquesta unitat, l'alumne/a:

1. Prepara arxius multimèdia per a la web, analitzant les seves característiques i manejant eines específiques.

- Reconeix les implicacions de les llicències i els drets d'autor en l'ús de material multimèdia.
- Identifica els formats dimatge, àudio i vídeo a utilitzar.
- Analitza les eines disponibles per generar contingut multimèdia.
- Empra eines per al tractament digital de la imatge.
- Utilitza eines per manipular àudio i vídeo.
- Fa animacions a partir dimatges fixes.

2. Integra contingut multimèdia en documents web valorant la seva aportació i seleccionant adequadament els elements interactius.

- Identifica i analitza les tecnologies relacionades amb la inclusió de contingut multimèdia i interactiu.
- Reconeix les necessitats específiques de configuració dels navegadors web per suportar contingut multimèdia i interactiu.
- Utilitza eines gràfiques per al desenvolupament de contingut multimèdia interactiu.
- Analitza el codi generat per les eines de desenvolupament de contingut interactiu.
- Afegeix elements multimèdia a documents web.
- Incorpora interactivitat a elements d'un document web.
- Comprova elements multimèdia i interactius en diferents navegadors.

1. Elements multimèdia al web: imatges, vídeo i àudio

Avui dia és impensable que una web no contingui elements multimèdia. Com a mínim s'espera trobar imatges, però el més habitual és que contingui tota una sèrie d'elements que poden incloure a més a més: animacions, sons, vídeos, aplicacions i jocs en 2D i 3D, i fins i tot s'està començant a experimentar amb aplicacions de realitat virtual (com es pot veure en aquesta URL de l'equip Mozilla VR: mozvr.com).

Realitat virtual a l'abast de tothom

A la conferència Google I/O 2014 es va regalar a tots els assistents un paquet de cartró retallat que permetia muntar un visor de realitat virtual introduint el mòbil com a pantalles, amb unes lents comunes que es poden trobar a qualsevol òptica i fent servir un parell d'imants com a botó.

Google va posar a disposició de tothom les especificacions i qualsevol pot descarregar les plantilles i fer-se el seu propi equip VR de cartó. Podeu trobar aquesta informació en el següent enllaç: www.google.com/get/cardboard.



Google Cardboard és el dispositiu de VR més econòmic que es pot trobar.
Font: othree (www.flickr.com).

Actualment, amb HTML5 s'ha volgut separar definitivament el disseny de l'estructura de les dades presentades als documents HTML. En aquesta versió trobem que les etiquetes que només servien per modificar la presentació han estat declarades obsoletes. Per altra banda, s'han afegit moltes més capacitats a través de les web API augmentant molt la potència d'aquest llenguatge de marques que, juntament amb CSS, ha arribat a desbancar completament altres tecnologies com Flash. Per aquesta raó, dominar aquestes tecnologies i el programari relacionat és indispensable per a qualsevol desenvolupador o dissenyador web.

Les **web API** són extensions d'HTML que afegeixen noves capacitats al llenguatge. Algunes són estàndards i es pot comptar a trobar-les a tots els navegadors moderns; alguns exemples són Web Audio; que permet fer una modificació avançada de la reproducció de sons, els Web Sockets, que permeten connectar amb servidors; WebGL, que és la implementació d'OpenGL per al web, etc. A aquestes funcionalitats noves s'accedeix a través del llenguatge JavaScript. Se'n pot trobar més informació a goo.gl/F0QSOQ.

1.1 Elements d'imatge per al web

El primer que cal distingir en parlar d'imatges per al web són els dos tipus fonamentals: mapes de bits i imatges vectorials. Atès que tant els orígens d'aquestes com la seva edició i ús són molt diferents, és necessari aprendre a distingir-los i a saber quan fer-ne servir l'un o l'altre.

Així doncs, segons el tipus d'imatge, aquestes podran trobar-se en diferents formats amb unes característiques determinades com pot ser el nombre de colors, el suport per transparències i/o animacions, compressió amb pèrdua o sense, etc.

Però el tipus no només determina els possibles formats, sinó també el programari que es pot utilitzar per editar-les i optimitzar-les, tot i que freqüentment les imatges vectorials s'acaben exportant com imatges de mapes de bits, el que obliga a conèixer el programari per tractar tots dos tipus.

1.1.1 Tipus d'imatges al web

Tothom ha sentit la frase “una imatge val més que mil paraules”, i això no és menys cert quan parlem de la web. Amb l'etiqueta es va revolucionar la forma de veure un document HTML i es va facilitar el desenvolupament de la web en altres àmbits com, per exemple, el comerç electrònic.

Us imagineu una botiga en línia sense fotografies dels seus productes? O la pàgina web d'un artista sense cap imatge de les seves obres? Fins i tot existeixen xarxes socials creades només en base a imatges, com per exemple Flickr (www.flickr.com), Pinterest (es.pinterest.com) o Instagram (www.instagram.com).

A l'hora de treballar amb imatges per a una pàgina web o altres aplicacions s'han de tenir en compte una sèrie de factors per utilitzar el format més adient, ja que segons el tipus d'imatge i la seva utilització el format a utilitzar serà un o un altre.

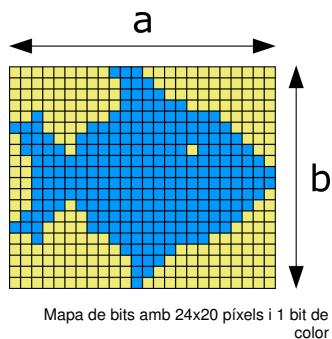
Es poden distingir dues categories d'imatges: per una banda, tenim els **mapes de bits** i, per una altra, les **imatges vectorials**. Encara que els que utilitzarem principalment són els primers, els segons també es fan servir molt i és important conèixer-les, atès que és possible treballar amb una imatge amb format vectorial i després exportar-la com a mapa de bits.

Mapes de bits

Un mapa de bits és una estructura de dades que representa una matriu de punts individuals, anomenats píxels, amb la informació de color de cada punt (vegeu la figura 1.1).

Un píxel es la unitat mínima que es pot representar en un monitor.

FIGURA 1.1. Exemple de mapa de bits



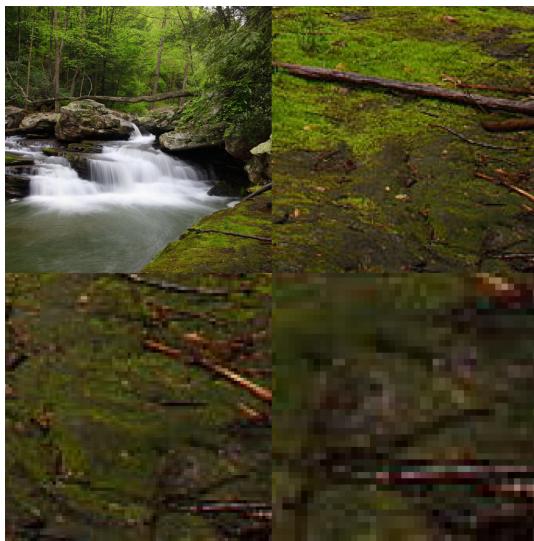
Per exemple, si tenim una imatge de 100x100 píxels amb 16 bits de color, el que tenim és una matriu de 100x100 que guarda la informació de 10.000 píxels, i cadascun d'aquests píxels tindrà un valor d'entre 0 i 65.535.

Com que aquest format ocupa molt espai, s'han desenvolupat diverses tècniques de compressió, i una de les més conegudes és JPEG. Aquest format permet fer servir transparències afegint als canals RGB (vermell, verd i blau) un canal més, anomenat alfa, que guarda la informació sobre la transparència de la imatge com una capa extra en escala de grisos. Això incrementa en 8 bits l'espai necessari per a cada nombre a la matriu del mapa de bits. Així, una imatge de 16 bits amb transparència, encara que només pot representar 65.535 colors, requerirà 24 bits per píxel.

L'inconvenient d'aquest tipus d'imatges respecte a les imatges vectorials és que si volem ampliar-les es perd qualitat i es fan cada vegada més borroses, perquè el que es fa és operar amb la informació de color que tenim, i encara que es poden suavitzar més o menys els colors, l'efecte és el mateix i es perd la nitidesa.

Una solució a aquest problema en l'entorn web la trobem en l'ús del mòdul **Media Queries** de CSS3, que ens permet ajustar diferents classes CSS segons les característiques del medi on es mostra la pàgina (vegeu la figura 1.2). D'aquesta manera, podem fer servir una imatge de fons amb més resolució en ordinadors d'escriptori i imatges molt més petites i lleugeres quan la mateixa pàgina es visualitza en un dispositiu mòbil.

FIGURA 1.2. Exemple d'una mateixa imatge a diferents escales



Fotografia amb diferents escales: superior esquerra 100%, superior dreta 500%, inferior esquerra 1000% i inferior dreta 2000%. Font: www.forestwander.com

Els casos d'ús més freqüents dels mapes de bits que trobem a Internet són:

- **Fotografies:** aquesta és l'única forma de mostrar-les; sempre que tinguem una fotografia la mostrarem en format de mapa de bits.
- **Icones:** els dos casos més comuns són la icona que es mostra a la pestanya del navegador, anomenada *favicon*, i la seva inclusió dins d'atles per

Mòdul de Media Queries

Forma part de CSS3 i ens permet adaptar la visualització a diferents condicions com pot ser l'amplada de la pantalla per mostrar o amagar diferents elements.

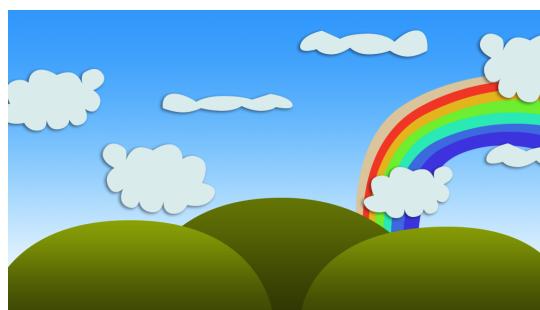
Generador de 'favicons' per a diferents dispositius

Per simplificar la feina a l'hora de generar *favicons* per a diferents dispositius podem fer ús de generadors com realfavicongenerator.net per obtenir totes les versions ràpidament.

optimitzar la descàrrega quan hi ha diferents símbols que volem utilitzar freqüentment a una pàgina, per exemple els símbols de “M’agrada” de Facebook.

- **Logotips:** encara que el més habitual és dibuixar els logotips com a gràfics vectorials a l'hora de fer-los servir, s'han d'exportar com a mapa de bits.
- **Imatges decoratives i de fons:** tot i que aquestes poden tenir com a origen tant un mapa de bits com una imatge vectorial, s'han d'exportar com a mapa de bits per fer-les servir (vegeu la figura 1.3). En alguns casos poden reemplaçar-se per alguns efectes amb CSS3, com els degradats.

FIGURA 1.3. Fons d'IOC-Puzzle



- **Sprites:** un *sprite* és una imatge o animació en dues dimensions que es fa servir en jocs (vegeu la figura 1.4). Per exemple, per mostrar una animació d'un personatge en moviment el que es fa és afegir una quantitat d'imatges que es van reemplaçant una darrere de l'altra a un ritme que enganya l'ull per crear la il·lusió d'animació. Un altre ús que es pot donar als *sprites* és combinar una sèrie d'imatges petites en un sol fitxer (per exemple, banderes o icones) i així minimitzar el nombre de peticions que cal fer al servidor per descarregar-les.

FIGURA 1.4. Cici the Cat caminant



Font LucarioShirona (www.deviantart.com)

Imatges vectorials

A diferència dels mapes de bits, les imatges vectorials treballen amb figures geomètriques com ara punts, línies, corbes, formes i polígons basades en expressions matemàtiques per definir les imatges (vegeu la figura 1.5).

Això permet mostrar el mateix resultat a qualsevol escala, ja sigui en el logotip d'una pàgina web, imprès en una samarreta o estampat en un globus aerostàtic.

FIGURA 1.5. Mostra d'una mateixa imatge vectorial amb diferents escales



Superior esquerra escala 100%, superior dreta escala 200%, inferior esquerra escala 400%, inferior dreta escala 800%.

Com podeu imaginar, aquest sistema només es pot aplicar a composicions senzilles, perquè no ofereix el grau de control a nivell de píxel que ofereix un mapa de bits i normalment el seu ús està limitat a figures, fonts, colors plans o amb degradats, però mai amb el nivell de qualitat d'una fotografia.

Tot i això, aquest format és ideal per a la realització de logotips, icones, imatges decoratives, fonts, gràfiques, etc. Precisament, en els últims anys, tant en el disseny de pàgines web com d'applicacions s'ha promogut l'ús de colors plans i imatges d'estil vectorial en lloc dels mapes de bits.

S'ha de tenir en compte que es pot treballar amb un programa de gràfics vectorials i després exportar aquestes imatges com a mapa de bits en diferents mides, perquè així sempre es treballa amb la màxima qualitat possible. Per exemple, per cobrir totes les opcions per afegir un *favicon* al vostre lloc web s'han de generar més de 10 imatges amb diferents mides; fent servir una imatge vectorial com a base es poden generar totes optimitzades, per a cada tipus, sense perdre qualitat.

HTML5 també incorpora suport nadiu per treballar amb imatges vectorials a través de l'etiqueta `<svg>`, que permet afegir elements vectorials directament a la pàgina.

```

1 <svg version="1.1"
2   baseProfile="full"
3   width="640" height="480"
4   xmlns="http://www.w3.org/2000/svg">
5
6   <rect width="100%" height="100%" fill="black" />
7
8   <circle cx="320" cy="240" r="160" fill="blue" />
9
10  <rect x="200" y="140" width="240" height="200" fill="grey"/>
11
12  <text x="320" y="240" font-size="100" text-anchor="middle" fill="white">IOC</
13    text>
14
15 </svg>

```

Mètodes alternatius per mostrar imatges SVG

En lloc d'afegir el codi dins de `<svg>` pot inserir-se en un fitxer fent servir l'etiqueta `<object data="image.svg" type="image/svg+xml">` o `<iframe src="image.svg"></iframe>`.

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/RryoKR i el resultat a la figura 1.6.

FIGURA 1.6. Resultat del codi SVG



Adobe Systems

Adobe és una multinacional creada el 1982 que ha desenvolupat, entre d'altres, el programari d'edició de gràfics més utilitzat tant per a mapa de bits com per a gràfics vectorials. Entre els seus productes podem trobar Adobe Photoshop i Adobe Illustrator.

Un cas especial de gràfics vectorials són les fonts. Originàriament, les fonts estaven formades per mapes de bits, però a partir de 1980 Adobe va introduir un tipus de fonts vectorials, la qual cosa permetia solucionar el problema de nitidesa en escalar les fonts i millorava la qualitat.

Ús de fonts com a gràfics vectorials: Font Awesome

Un ús molt interessant de les fonts és el que fa **Font Awesome** ([fortawesome.github.io/Font-Awesome](https://fontawesome.github.io/Font-Awesome)), que carrega una font amb més de 500 icones i ens permet incrustar-les a les nostres pàgines fent servir CSS.

Molts llocs web fan servir aquesta llibreria de manera que l'efecte que es produeix en fer servir aquestes icones és d'homogeneïtat.

Els gràfics vectorials es fan servir en els següents casos, tenint en compte que generalment cal exportar-los com a mapes de bits amb la mida desitjada:

- **Logotips:** la creació dels logotips és interessant fer-la com a gràfics vectorials i després exportar-los al format que ens requereixin. Per exemple, en una impremta segurament ens demanaran el disseny en format vectorial, però per a una pàgina web s'exportarà com a mapa de bits.
- **Motius decoratius:** a causa de la naturalesa dels gràfics vectorials, és molt més fàcil generar patrons (com el que es pot apreciar a la figura 1.7) o imatges a partir de fòrmules matemàtiques amb una gran precisió.

FIGURA 1.7. Aplicació del patró



Font: AssEyeDee (www.deviantart.com)

- **Cartells:** encara que generalment no es farà servir únicament el format vectorial, és interessant aplicar-lo sempre que sigui possible, ja que ens assegura que encara que les mides del cartell siguin molt grans la qualitat es mantindrà intacta (vegeu la figura 1.8).

FIGURA 1.8. Cartell: Federal Theatre.
Marionette Theatre presents "RUR"



Font: Charles Verschuuren (commons.wikimedia.org)

- **Icones:** generalment, les icones estan compostes amb colors plans i formes senzilles, per la qual cosa el seu disseny fent servir aquestes imatges és ideal. Això inclou els *favicons* que es mostren a les pestanyes dels navegadors.
- **Imatges per a jocs web o mòbils:** encara que històricament els jocs en 2D han fet servir sempre mapes de bits (i actualment estan de moda els jocs *retro* que exageren aquest aspecte pixelat), s'han produït molts jocs web i per a dispositius mòbils amb tots els seus continguts dissenyats com a imatges vectorials (vegeu la figura 1.9).

FIGURA 1.9. Angry Birds Friends



Els jocs de la sèrie Angry Birds fan servir imatges vectorials per als elements del joc i mapes de bits per a la interfície.

2D Game Art for programmers

Al blog de Chris Hildenbrand podeu trobar una gran quantitat de tutorials mostrant com crear art per a jocs amb programari lliure:

www.2dgameartguru.com.

L'ús d'imatges vectorials en aquest àmbit és molt interessant, ja que permet generar tots els gràfics per als diferents dispositius sense gaire esforç. Això facilita un desenvolupament molt més ràpid quan l'objectiu són plataformes amb resolucions tan diferents com un mòbil, una tauleta o un televisor de 52”.

Especialment en el cas de jocs com **Angry Birds**, s'aprofiten molt millor els gràfics vectorials, perquè la major part de les animacions només treballen amb rotacions, canvis de posició i canvis d'escala que s'apliquen sobre la imatge vectorial, en lloc de necessitar una pila de mapes de bits per simular totes les animacions.

1.1.2 Formats d'imatges

Existeixen molts formats diferents d'imatges, però els més usuals són GIF, JPEG i PNG. Un nou format, WebP, que encara que no és suportat per tots els navegadors, actualment té unes característiques que el fan molt interessant (vegeu la taula 1.1).

TAULA 1.1. Comparativa de formats

Característica	GIF	JPEG	PNG	WebP
Nombre de colors	8 bits	24 bits	24 i 32 bits	24 bits
Compressió	Sense pèrdua	Amb pèrdua	Sense pèrdua	Amb pèrdua i sense
Transparència	Limitada	No	Sí	Sí, fins i tot amb pèrdua
Es pot animar	Sí	No	No	Sí

Comparativa dels formats d'imatges més coneguts de tipus mapa de bits

També podem trobar dos tipus especials: el format SVG i l'ús d'imatges com a Base64, que és una manera una mica particular d'afegir imatges sense haver de descarregar-les separadament.

Hi ha moltes característiques comunes als formats que cal conèixer i saber distingir:

Es pot consultar la llista completa de tipus *mime* assignats a: goo.gl/BmtqDd.

- **Tipus *mime*:** el tipus *mime* d'un format s'inclou a la capçalera d'aquests fitxers i és necessari per poder interpretar de quin tipus es tracta en alguns casos, com per exemple quan l'incrustem com a Base64. Està format per un tipus, un subtípus i uns paràmetres opcionals, encara que en el cas de les imatges només trobem el tipus i subtípus, per exemple “image/png”.
- **Compressió amb pèrdua:** els formats amb aquest tipus de compressió poden ocupar menys espai, però a costa de la qualitat de la imatge. Com major sigui la compressió, menor serà la seva qualitat respecte a l'original.
- **Compressió sense pèrdua:** la mida d'aquests formats és més gran que els anteriors, però la imatge és idèntica a l'original.

- **Entrellaçat:** l'entrellaçat és una opció suportada per gairebé tots els formats que permet començar a visualitzar la imatge abans que estigui completament descarregada, com si fos una mena de vista prèvia.
- **Metades:** encara que gairebé tots els formats admeten metades, s'ha de tenir en compte que no sempre es conserven en passar d'un format a un altre. Per exemple, en passar de JPEG a PNG és possible que es perdin totes les metades referents a la localització on s'ha pres una foto o el nom de l'autor. S'ha de consultar el programari amb el qual es fa la conversió per veure quines opcions ens ofereix per conservar o reescriure aquestes metades.

Podeu trobar més informació sobre l'etiqueta `img` en el següent enllaç: goo.gl/XydC83.

Principalment, aquestes imatges són incrustades al web utilitzant l'etiqueta `img` d'HTML, com es pot veure en el següent exemple:

¹ ``

Com podeu veure, per utilitzar una imatge només cal indicar el seu origen (propietat `src`) i un text alternatiu (propietat `alt`) que es mostra quan no es pot mostrar la imatge.

GIF

El format GIF va ser un dels primers formats emprat en Internet. Té dues característiques principals: per una banda, permet definir un color com a transparent, i per una altra, permet crear animacions.

A causa de les limitacions de colors i transparències, ha caigut en desús a favor del format PNG, encara que per afegir animacions simples continua sent una bona opció, com es pot veure per exemple a Facebook, que inclou l'opció de fer servir GIF als seus comentaris i entrades.

Quan es treballa amb logotips, com que normalment inclouen pocs colors, és factible fer servir aquest format, encara que és més adient el format PNG. En canvi, si fem servir aquest format per a fotografies ens trobarem amb una pèrdua de qualitat molt gran, com es pot apreciar a la figura 1.10.

FIGURA 1.10. Conversió de fotografia a format GIF



Font: Matteo Paganelli (unsplash.com)

Podeu trobar més informació sobre el format GIF en el següent enllaç: https://es.wikipedia.org/wiki/Graphics_Interchange_Format.

Llocs web especialitzats en GIF animats

Un d'aquests llocs és giphy.com, on es poden trobar multitud de GIF animats en què s'aprofiten captures de vídeos convertides en animacions.

JPEG

Podeu trobar tota la informació sobre el format JPEG en el següent enllaç: ca.wikipedia.org/wiki/JPEG.

Què són els artefactes?

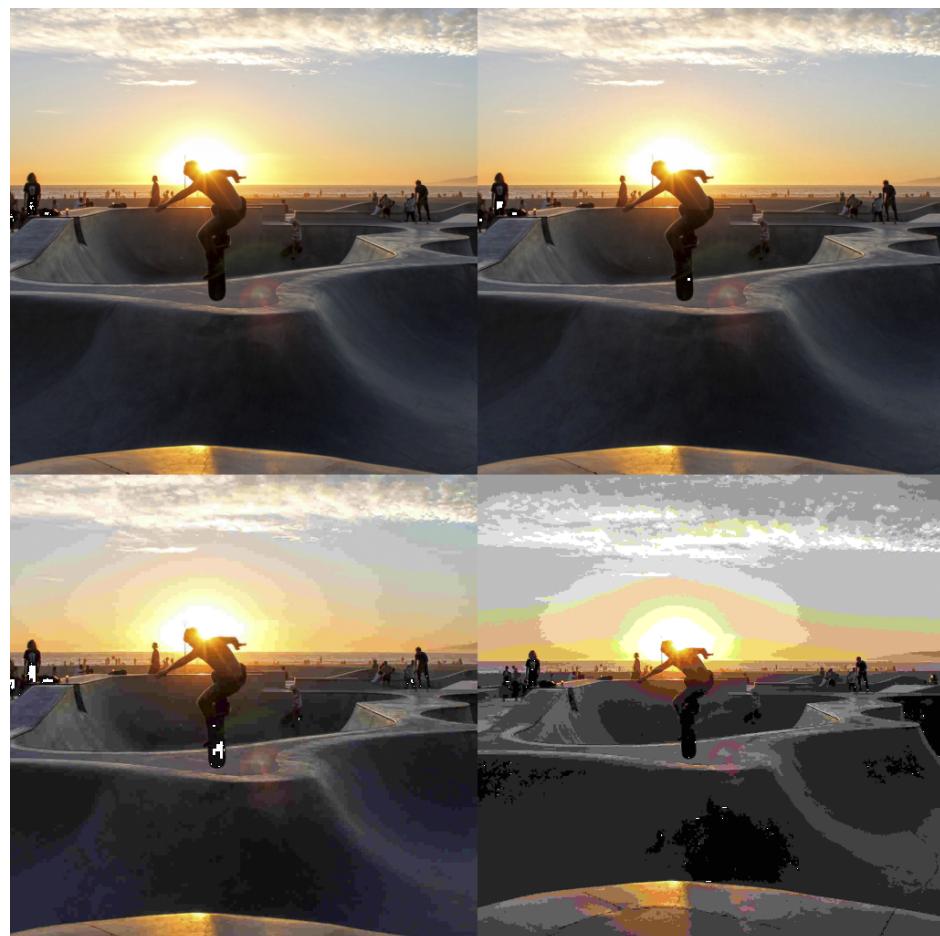
S'anomena *artefactes* als punts de la imatge on, en aplicar els algorismes de compressió, surt distorsionada. Frequentment es troben a les vores on hi ha un canvi important de color o contrast.

Normalment, les càmeres digitals guarden les fotografies amb aquest format per estalviar espai.

Aquest format és ideal per a fotografies perquè redueix molt la mida dels fitxers, i la pèrdua de qualitat pot ser inapreciable. En canvi, no és indicat per a imatges amb colors plans, perquè es generen artefactes a les zones properes on es produeix un canvi de colors.

S'ha de tenir en compte que, si treballeu amb aquest format, cada vegada que generieu una nova imatge la qualitat d'aquesta serà pitjor, ja que per cada vegada que s'aplica la compressió la pèrdua serà major com es pot apreciar a la figura 1.11. Per aquesta raó és important, sempre que sigui possible, **treballar amb les imatges originals** o exportades amb un format **sense pèrdua**, i només exportar a format JPEG la imatge final.

FIGURA 1.11. Comparativa de qualitat d'imatges en JPEG



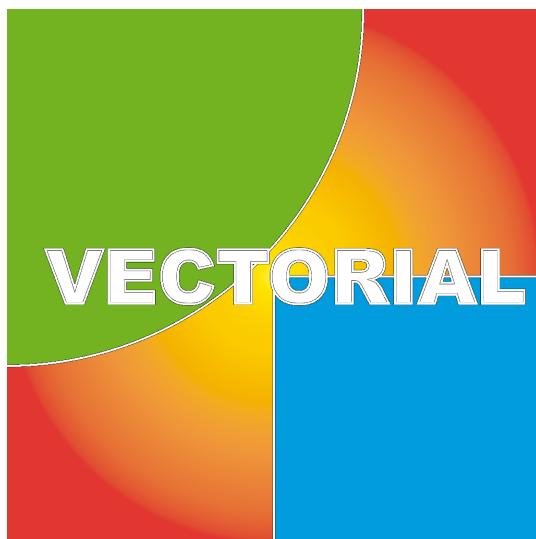
Superior esquerra 100% de qualitat, superior dreta 50%, inferior esquerra 10%, inferior dreta 0%. Font: Matteo Paganelli (unsplash.com)

PNG

Podeu trobar més informació sobre el format PNG en el següent enllaç: es.wikipedia.org/wiki/Portable_Network_Graphics.

Aquest format substitueix en gairebé tots els casos els GIF, perquè permet treballar amb més color i més detall a les transparències, i no hi ha pèrdua de qualitat a les imatges. És el més indicat quan es treballa amb imatges amb colors plans, ja que la qualitat és perfecta (com es pot apreciar a la figura 1.12) i, gràcies a com es comprimeixen les dades en treballar amb grans grups de píxels del mateix color, redueix molt la mida dels fitxers.

FIGURA 1.12. Imatge vectorial exportada com a PNG



WebP

Aquest és un format desenvolupat per Google que inclou les mateixes característiques que altres formats vistos anteriorment i els millora produint imatges amb una mida inferior i la mateixa qualitat, permetent l'animació d'imatges amb 24 bits de color i fer servir el canal alfa amb un algorisme de compressió amb pèrdua.

El format **WebP** no té un *mime type* oficial.

Es pot trobar informació més detallada sobre el format WebP a la web de Google Developers:
developers.google.com/speed/webp.

El problema amb aquest format és que, almenys de moment, només és compatible amb Google Chrome, Opera i dispositius amb Android 4.0 o superior, perquè altres companyes desenvolupadores de navegadors no veuen clar que el format WebP superi per un marge significatiu el format JPEG.

Un altre inconvenient és que el programari de disseny gràfic no inclou normalment l'opció d'exportar en aquest format nativament, i s'han de fer servir connectors de tercers per afegir aquesta opció.

SVG

Aquest és un format de gràfics vectorials. Al contrari que els altres formats vistos anteriorment, que guarden la informació de les imatges segons el color del píxel a una posició determinada, en SVG el que es guarda són les formes geomètriques i les seves posicions. Les seves característiques són les següents:

- Pot mostrar tots els colors que suporti el navegador.
- Compressió sense pèrdua fent servir l'algorisme *gzip*.
- Suporta animacions via **JavaScript** i **CSS**.
- És possible aplicar transparències a elements concrets.

- El tipus *mime* corresponent és **image/svg+xml**.

Exportar imatges vectorials a format SVG

Cal tenir en compte que el programari d'edició d'imatges vectorials permet exportar aquests dissenys més complexos a fitxers amb extensió .svg i .svgz.

Les imatges es poden incrustar directament al fitxer HTML fent servir l'etiqueta <svg> o poden guardar-se en un fitxer amb extensió .svg.

També és possible animar aquests dibuixos fent modificacions a les posicions dels elements fent servir **JavaScript** i **CSS**, encara que no tots els navegadors apliquen aquestes transformacions correctament, ja que alguns directament ignoren les instruccions i altres consideren el punt d'origen de manera diferent.

Base64

Hi ha algunes situacions en les quals pot interessar obtenir una representació d'una imatge que es pugui enviar o incrustar directament a un fitxer HTML o CSS, evitant haver d'afegir aquesta com un fitxer independent. Per exemple, per afegir emoticons a un programari de missatgeria instantània, per crear una plantilla de correu electrònic que només utilitzi un únic fitxer HTML o per crear classes CSS per mostrar banderes com icones sense haver de llegir múltiples fitxers d'imatges.

Per solucionar aquests problemes, entre d'altres, disposem de Base64. Es tracta d'un sistema de codificació que ens permet convertir les dades binàries en un sistema de 64 caràcters que poden ser impresos i transmesos sense problemes a través de la xarxa. Per exemple, el codi binari que forma el *favicon* de l'IOC pot convertir-se a Base64 i obtindríem la següent cadena:

```
1  AAABAAEAEBAAAAIABoBAAAFgAACgAAAAQAAAIAAAAAEIAAAAAAAAAAAAAAAAAAAAAAA
2  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYYoAAWx9AAAAAAAAAAAAAAAAAAAAAA
3  AAAAAAAAAAAAAAAAAAAAAAAAAA2JsANNqaAM7ZnAC/1
4  pgAJgAAAAAAAAAAAAAA
5  AAAAAAAAAAAAAAAAAAAAAAAADWmAAW2JsAotqcAP/anAD/2pwA/9qcAP/
6  ZmgCR05sGDgAA
7  AAAAAAAAAAAAAAAAAAAAAAA+JAATYmgB32pwA9dqcAP/anAD/2pwA/9qcAP/anAD/2pwA
8  /9qcA0/YmgBo0IRSAAAAAAAAAAAAAANmcAfZmwDk2pwA/9qcAP/anAD/2pwA/9qcAP/anAD/2
9  pwA
10 /9qcAP/anAD/2pwA/9mbAnAAAAAAADcnQAR2ZoA/9qcAP/anAD/2pwA/9qcAP/anAD/2
11 pwA
12 /9qcAP/anAD/2pwA/9qcAP/ZmwD/AAAAAAAAAAAAA3J0AEdmaAP/anAD/2pwA/9qcAP/anAD/2
13 pwA
14 /9qcAP/anAD/2pwA/9qcAP/anAD/2pwA/9qcAP/ZmwD/AAAAAAAAAAAAA3J0AEdmaAP/anAD/2
15 pwA
16 /9qcAP/anAD/2pwA/9qcAP/anAD/2pwA/9qcAP/anAD/2pwA/9qcAP/anAD/2psA/
17 NmbAIzXmwALAAAAAAAAAAAAAA
18 AAAAAAAAAAAAAAAACMAAAAAAAAAAAAAAAACNeaAC/
YmwDH2ZoAuNiaACMaaaaaaaaaaaaaaaaaaaaaaa
19 AAAAAAAAAAAAAAA2aEAAgAAAAAAAAAAAAAA
```

Podeu trobar un codificador/descodificador de Base64 en el següent enllaç:
www.base64encode.org.

La codificació Base64 no inclou salts de línia i per tant, s'han d'eliminar aquests per fer funcionar els exemples.

19 AAAAAAAAAAAA//8AAP5/
AAD4HwAA8A8AMADAADAAwAAwAMAAMADAADAAwAAwAMAAMADAADAAwAA8A8A
20 APg fAAD+fwAA//8AAA==

Llavors, aquesta informació es pot incloure com a font d'un fitxer d'imatges, fent servir **l'esquema data URI** en lloc dels esquemes HTTP o HTTPS.

Vegeu un exemple de com es pot incrustar la imatge directament a una etiqueta `` o a un fons definit amb CSS; per exemple, el codi per incrustar el *favicon* de l'IOC com a imatge fent servir l'etiqueta `` seria:

```
1 
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/xZjRgv.

Encara que afegir imatges d'aquesta manera té l'avantatge de disminuir el nombre de peticions HTTP que s'han de realitzar per carregar la pàgina, aquest sistema representa una sèrie d'inconvenients:

- El codi no és gens llegible.
- La codificació augmenta la mida del fitxer un 33% aproximadament respecte al que ocupa l'original en binari.
- Aquest sistema no és suportat per tots els navegadors.
- Les imatges afegides d'aquesta manera requereixen més temps per ser processades que en el format original.

Data URI

L'esquema *data URI* permet incloure dades *in-line* a les pàgines web com si fossin recursos en línia, afegint la informació del fitxer en forma literal en lloc de descarregar-lo.

1.1.3 Programari per crear i processar imatges

Una vegada vistos els diferents formats, és hora de veure com es poden crear i formatar imatges pròpies. Tot i que hi ha eines per modificar les imatges des de la línia de comanda, ens centrarem en programari que compta amb una interfície gràfica.

A Internet es poden trobar tota mena d'imatges, però no sempre trobareu les que necessiteu o bé no les podeu utilitzar tal com estan, ja sigui perquè les seves llicències no ens ho permeten, perquè necessiteu una imatge molt concreta o perquè s'ha d'optimitzar la seva mida. En altres ocasions, les imatges us seran proporcionades pel mateix client que us hagi encarregat desenvolupar la web, o fins i tot pot tractar-se d'imatges pròpies.

Sigui com sigui, és molt habitual que no pugueu fer servir les imatges en la seva forma original i hagiu de retocar-les i optimitzar-les abans de fer-les servir, cosa que no és possible fer a través d'HTML.

Encara que habitualment les tasques que realitzareu amb el **programari d'edició** d'imatges estaran centrades en l'optimització i el canvi de mida dels recursos gràfics, és imprescindible conèixer les eines bàsiques d'aquests tipus de programari per poder portar a terme, com a mínim, algunes tasques de dibuix i retoc simples, com per exemple crear un logotip a partir dels materials d'una empresa o personalitzar icones que es faran servir a la pàgina.

Exemple de processament d'imatges per a web

Us han encarregat desenvolupar la pàgina web d'una fusteria i el client us ha fet arribar totes les imatges de què disposa i que ja han fet servir en el seu catàleg en paper.

- El logotip de l'empresa us l'ha passat com a imatge vectorial.
- Les fotografies del catàleg són JPEG amb màxima qualitat, i cada fotografia ocupa més de 10 MB.

En el cas del logotip, abans de poder utilitzar-lo l'haureu de convertir en mapes de bits amb diferents mides segons on el necessiteu fer servir (favicon, logo de l'empresa, marca d'aigua sobre fotos, etc.) i les fotografies les haureu de redimensionar per ajustar-les a una mida adequada per veure-les en un monitor i exportar-les amb un nivell de compressió més alt per reduir dràsticament el seu pes.

Per fer aquests canvis necessitareu un programari d'edició gràfica que us permeti treballar amb mapes de bits o imatges vectorials, segons el tipus d'imatge que tracteu.

Tant en el cas de programari gratuït com en el de pagament, es poden trobar multitud de connectors (*plugins* en anglès) que augmenten la capacitat d'aquests programes. Per exemple, un tipus de connectors molt habituals en el programari d'edició de mapes de bits són els que afegeixen nous filtres i els que afegeixen la capacitat d'exportar en nous formats.

Recentment, la família de S.O. d'Apple han modificat els seus noms quedant així:

- macOS
 - iOS
 - tvOS
 - watchOS
-

Entre el programari d'edició d'imatges en format de mapa de bits trobem moltes opcions, però aquestes dues són les més populars:

- **GIMP**: és un programari gratuït multiplataforma que forma part del projecte GNU. Permet obrir gairebé tots els tipus d'imatges de tipus mapa de bits i exportar en els formats més habituals.
- **Adobe Photoshop**: en aquest cas, es tracta d'un programari privatiu líder en el sector només disponible per a les plataformes Windows i macOS. Permet treballar amb un nombre menor de formats i exportar només als més habituals (GIF, PNG, JPEG i SVG). Destaca la gran quantitat de connectors de tercers disponibles, una interfície molt intuitiva i la possibilitat de treballar amb formes vectorials.

Quant a l'edició d'imatges vectorials la selecció és molt més limitada i, com en el cas anterior, us mostrem dos programaris molt coneguts:

- **InkScape**: igual que GIMP, aquest és un programari gratuït multiplataforma que permet obrir molts formats d'imatges vectorials, guardar-les amb format SVG i SVG comprimit i exportar en format PNG.
- **Adobe Illustrator**: com és el cas d'Adobe Photoshop, aquest programari és privatiu, està limitat als usuaris de Windows i macOS. La seva interfície és molt intuitiva, però els formats que permet obrir són molt més limitats que els que suporta InkScape. Aquest és el més utilitzat en el disseny d'imatges vectorials i destaca per poder exportar directament en formats suportats per les impremtes.

Un avantatge del **programari gratuït** és que per desenvolupar tasques puntuals com són el canvi de format o optimitzar una imatge no necessitem més; per altra banda, professionalment són més valorades les competències adquirides amb el programari privatiu.

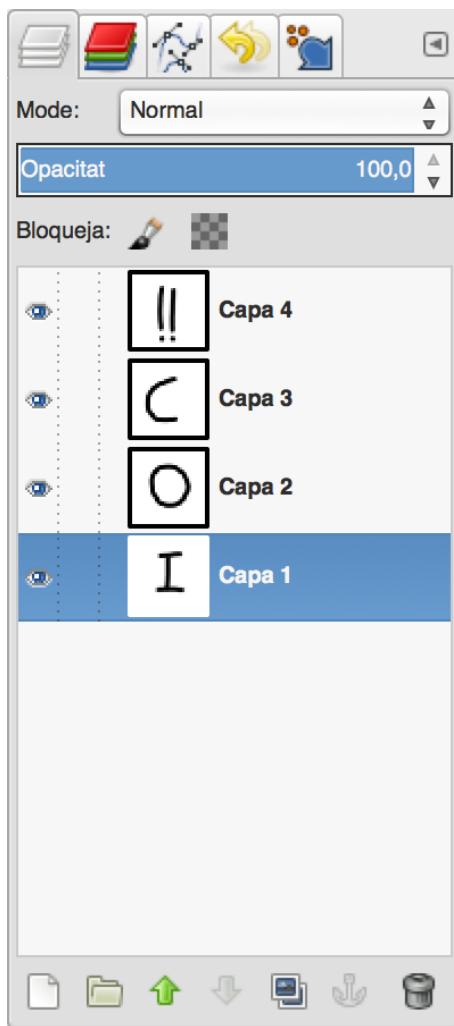
Funcionalitats del programari d'edició gràfica

El tractament en profunditat d'un programari d'edició gràfica escapa a l'abast d'aquests materials; per tant, ens centrem a mencionar algunes de les opcions més comunes que podem trobar en aquest tipus de programari i particularment en **GIMP**, per ser multiplataforma i gratuït.

Hi ha alguns conceptes que cal tenir clars abans de descriure les eines:

- **Diferència entre guardar i exportar**: cada programari fa servir un format propi on guarda informació extra sobre la imatge; aquest format no està optimitzat per fer-se servir en altres aplicacions, així que tots incorporen una opció per exportar a diferents formats (PNG, JPEG, etc.).
- **Tapís**: és l'àrea de dibuix; depenent del programari, es poden moure elements fora del tapís, però continuen formant part de la imatge, encara que a l'hora d'exportar-la només s'exporta el que estigui dins del tapís.
- **Capa**: tot el programari d'edició modern incorpora un sistema de capes. Cada capa és com un full transparent, de manera que si dibuixem a la capa superior se superposa a les altres. Aquestes capes també poden ser opaques o es poden aplicar diferents **modes de fusió** per aconseguir diferents efectes, per exemple invertir els colors, aclarir-los o enfosquir-los.

Com podeu veure a la següent figura figura 1.13, des del panell de la pestanya de capes a GIMP es poden crear noves capes, duplicar les actuals, canviar la posició, agrupar-les en carpetes, eliminar-les, modificar la visibilitat i canviar el seu nom, entre altres opcions.

FIGURA 1.13. Propietats de capes a GIMP

L'ús de les **capes** és molt important perquè permet organitzar la distribució de la imatge fàcilment, aplicar els modes de fusió i ajustar l'opacitat.

Error amb la selecció el·líptica

En alguns sistemes operatius amb la versió 2.8.14 de GIMP, l'eina de selecció el·líptica no funciona correctament, ja que no permet treballar amb la selecció.

Dins de GIMP podem veure el nom de cada eina deixant el cursor sobre l'eina uns segons. En els casos en què el nom en anglès és molt més conegut s'ha afegit a continuació el nom en català.

En general, tant les **seleccions** com totes les **accions** que portem a terme sobre una imatge (dibuixos, filtres, etc.) només s'apliquen a la capa activa. És a dir, si tenim tres capes i tenim seleccionada la capa superior, les seleccions afectaran només aquesta capa i no les inferiors. Per exemple si omplim la selecció amb un degradat aquest s'aplicarà a la capa superior, i les altres dues continuaran sense canvis. Si volem aplicar una selecció a una capa diferent hem de clicar sobre la capa on volem que s'appliqui, llavors aquesta passarà a ser la capa activa i qualsevol acció l'affectarà.

Pràcticament totes les eines de selecció tant de GIMP (que podeu trobar a la taula 1.2) com d'altres programaris d'edició ofereixen les mateixes opcions:

- **Suavitzat (antialiasing)**: la selecció es fa de forma suavitzada, de manera que les vores es fan transparents progressivament i el canvi de colors no és directe.
- **Vores arrodonides (feathering)**: les accions sobre la selecció s'apliquen de manera molt suavitzada fora dels límits de la selecció.
- **Relació d'aspecte**: per a les eines de selecció de tipus poligonal es pot establir una relació d'aspecte, per exemple 1:1, per aconseguir cercles o quadrats perfectes.
- **Modes de selecció**: la selecció pot reemplaçar la selecció actual, afegir-la, restar-la o aplicar una intersecció entre la selecció nova i les antigues.

TAULA 1.2. Eines de selecció

Eina	Descripció
Eina de selecció rectangular	Aquesta eina permet crear seleccions amb un rectangle que es pot modificar arròssegant les cantonades o vores.
Eina de selecció el·liptica	Similar a l'anterior però amb forma el·liptica, també es pot modificar arròssegant les cantonades o vores.
Eina de selecció lliure (<i>lasso</i>)	Aquesta eina permet crear una selecció fent clics amb el ratolí per marcar punts entre línies o, si deixem el botó polsat, arròssegant per dibuixar l'àrea a seleccionar.
Eina de selecció de regió continua (<i>magic wand</i>)	Amb aquesta eina se selecciona automàticament tot el que hi hagi dintre d'una àrea, detecta les vores segons la diferència de colors.
Eina de selecció de per color	Aquesta eina permet fer una selecció de tots els elements del dibuix amb el mateix color o similar.
Eina de tisores intel·ligents	Permet fer una selecció clicant punts propers a la forma que volem retallar, i aquesta s'ajusta automàticament.

Les seleccions ens permeten modificar els continguts i limitar els efectes d'altres eines, per exemple amb les eines d'ompliment només s'omplirà la selecció si es troba alguna i tampoc no es podrà pintar fora de la selecció ni esborrar.

També ens permeten copiar elements i enganxar-los en la mateixa capa o en unes altres, moure aquests elements i aplicar les transformacions que apareixen a la taula 1.3.

Per **copiar i enganxar** elements seleccionats entre capes només hem de seleccionar el que volem copiar, anar a *Edita / Copia*, clicar sobre la capa on volem enganxar i a continuació a *Edita / Enganxa*. Quan fem això ens quedarà sobre la capa una nova capa especial amb el nom *Selecció flotant*. Si fem clic amb el botó secundari del ratolí i seleccionem l'opció *Fixa la capa* s'afegeirà a la capa que surt just a sota, però si li donem a l'opció o a la icona de *Capa nova* es crearà una nova capa amb aquest element enganxat.

TAULA 1.3. Eines de transformació

Eina	Descripció
Eina gira	Ens permet girar una imatge en la direcció de les busques del rellotge o al contrari.
Eina escala	Permet escalar la imatge per fer-la més gran o més petita.
Eina inclina	Amb aquesta eina s'inclina la imatge cap a la dreta o cap a l'esquerra des d'un punt vertical central.
Eina perspectiva	Aquesta transformació permet aplicar un efecte de perspectiva sobre la capa o selecció.
Eina capgira	Permet capgirar la capa o la selecció horitzontal o verticalment com si fos un reflex en un mirall
Transformació de la regió	Amb aquesta creem una selecció de forma poligonal tancada i llavors podem modificar els vèrtexs per distorsionar la imatge.

La majoria d'aquestes transformacions afectaran tota la capa o, si existeix una selecció, només la part seleccionada de la capa. D'aquestes, les més usades són **girar, escalar i capgirar**.

Motius, patrons i pinzells

El programari de disseny permet dibuixar amb formes que poden emular el traç d'un pinzell real, taques de tinta, herba, flors, motius de fantasia, etc. A aquests elements habitualment se'ls coneix com *pinzells*. De manera similar, per omplir seleccions es poden fer servir patrons o motius, que són formes que es poden repetir sense que es pugui diferenciar on comencen i on acaben.

Altre grup d'eines molt utilitzat són les eines de dibuix que podeu trobar a la taula 1.4. Cal destacar que aquestes permeten fer servir un *pinzell* en lloc d'un color pla, però en el cas del *llapis* aquest no respecta les transparències i per tant no mostrarà un aspecte suavitzat.

TAULA 1.4. Eines de dibuix

Eina	Descripció
Eina pinzell	Amb aquesta eina es pot pintar amb un <i>traç suau</i> .
Eina llapis	Aquesta eina també pinta però només amb el color del front i l'efecte queda molt pixelat; es nota clarament quin píxel està pintat i quin no, es pot dir que és un <i>traç dur</i> .
Aerògraf	Aquesta eina funciona com un aerògraf: com més temps passem amb el botó polsat major serà l'opacitat del color, i si fem una passada ràpida només s'aplicarà una mica de color.
Eina goma d'esborrar	Amb aquesta eina podem esborrar qualsevol element de les imatges; inclou les opcions per fer una esborrada dura o suau.

Fixeu-vos que cadascuna d'aquestes eines té diferents opcions que modifiquen la forma com es pinta o s'esborra amb aquestes; per exemple, l'opacitat és comú a totes, però d'altres com la velocitat i el flux només es mostren per a l'aerògraf.

GIMP no té la capacitat d'altres programaris de treballar amb formes com es pot veure en la taula 1.5. El més semblant a fer un dibuix vectorial és traçar el contorn amb aquesta eina i després aplicar una d'aquestes dues opcions:

- **Selecció a partir d'un camí:** crea una selecció que podem omplir amb l'eina **cubell de pintura** o l'eina **de degradat**.
- **Pinta el camí:** aquesta opció permet dibuixar una línia que recorre tot el camí.

TAULA 1.5. Eines de traç

Eina	Descripció
Eina de camins	Permet fer un traç poligonal o amb corbes i transformar-lo en una selecció o pintar una línia.

S'ha de tenir molt en compte que una vegada s'aplica qualsevol d'aquestes opcions el resultat és independent del camí; si el camí es modifica no afectarà pas ni la selecció ni la línia pintada.

Un ús força freqüent que es dóna a les eines que podeu veure a la taula 1.6 és omplir una capa superior i canviar el seu mode de fusió a un altre, de manera que el patró o degradat se superposa sobre les capes inferiors.

TAULA 1.6. Eines d'emplenament

Eina	Descripció
Eina cubell de pintura	Omplie la selecció o capa amb el color del front o el patró seleccionat.
Eina degradats	Omplie la selecció o capa amb el degradat seleccionat.

GIMP inclou també un lloc d'eines genèriques que podeu trobar a la taula 1.7, aquestes (o altres equivalents) es poden trobar en tot el programari d'edició de mapes de bits.

TAULA 1.7. Altres eines

Eina	Descripció
Eina capturador de color	Serveix per seleccionar el color exacte d'un punt de la pantalla.
Eina ampliació (zoom)	Amplia la vista de la imatge, és a dir, com la veiem nosaltres, no la seva mida real.
Eina compàs	Serveix per mesurar la distància entre dos punts
Eina mou	Mou la selecció o capa seleccionada.
Eina escapça	Permet seleccionar una àrea, i en fer clic a sobre l'escapça, ajustant la mida de la imatge a la d'aquesta àrea seleccionada.

Degradats

Un degradat és una escala entre dos o més colors que comença en un i acaba en un altre. Generalment es poden incloure també nivells d'opacitat a cadascun d'aquests colors.

Cal destacar que, com tot el programari d'edició, GIMP us ofereix una bona col·lecció de filtres que permeten modificar les vostres imatges per afegir efectes especials, convertir fotografies en còmics, difuminar les imatges, eliminar o afegir soroll, etc.

Filtres a GIMP

Per veure una llista completa dels filtres que us ofereix GIMP podeu consultar el següent enllaç:
docs.gimp.org/ca/filters.html.

1.1.4 Optimització d'imatges per a la web

Optimització d'imatges a Google Developers

En aquest enllaç de la web de Google Developers podeu trobar més informació sobre l'optimització d'imatges: goo.gl/jp8vBy.

Adobe Flash

Flash va ser una tecnologia molt utilitzada per afegir elements multimèdia al web des de mitjan anys noranta que va començar a ser reemplaçada per HTML5 a partir del 2010.

Actualment, per millorar la presentació de les pàgines webs només cal utilitzar CSS i HTML, ja que els navegadors moderns han deixat de donar suport a connectors de tercer com Flash i Java.

Hi ha dos factors determinants per optimitzar les imatges per a la web: un és el format d'imatge i el seu nivell de compressió, i l'altre és la mida d'aquesta.

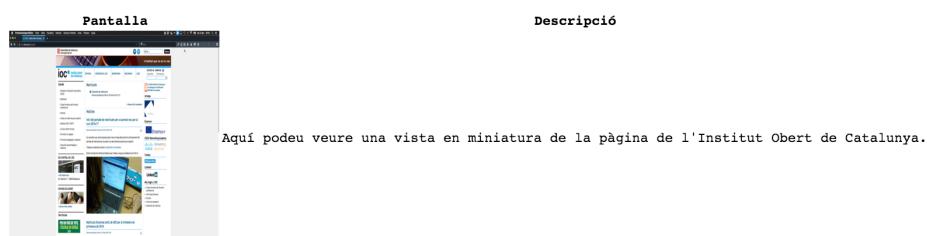
S'ha de tenir molt en compte com es farà servir la imatge, perquè encara que aquesta es pot engrandir o reduir fent servir regles CSS, no té gaire sentit fer servir una imatge gegantina més gran que la pantalla per mostrar una vista en miniatura d'un producte. Fixeu-vos en el següent exemple, on fem servir una imatge de 1920x1080 quan hauria sigut suficient amb una imatge de 200x200:

```

1 <style>
2   img {
3     width:200px;
4     height:200px;
5   }
6 </style>
7 <table>
8   <tr>
9     <th>Pantalla</th>
10    <th>Descripció</th>
11   <tr>
12   <td></td>
14   <td>Aquí podeu veure una vista en miniatura de la pàgina de l'Institut
15     Obert de Catalunya.</td>
16 </tr>
</table>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/xVJQrL, i el resultat a la figura 1.14.

FIGURA 1.14. Visualització de l'exemple de vista en miniatura



Aquesta imatge pesa més de 730 KB, però tenint en compte les necessitats reals d'utilització s'hauria pogut reduir a menys de 30 KB. No tan sols el pes no es troba optimitzat, sinó que la imatge es troba distorsionada perquè no s'han conservat les proporcions.

En la taula 1.8 podeu veure la diferència de pes d'una fotografia d'alta resolució en diferents formats i nivells de compressió.

TAULA 1.8. Comparativa de formats per a fotografia en alta resolució

Imatge	Mida	Format	Compressió	Pes
Skate	4.022x2.681	PNG		6.469 KB
Skate	4.022x2.681	JPEG	100%	3.473 KB
Skate	4.022x2.681	JPEG	50%	382 KB
Skate	4.022x2.681	JPEG	10%	176 KB
Skate	4.022x2.681	JPEG	0%	141 KB

S'ha de tenir en compte que el format **GIF** només suporta 256 colors, mentre que la resta treballen amb 24 bits.

En la taula 1.9 podeu veure les dades de la mateixa imatge però redimensionada per omplir una pantalla Full HD. Comproveu amb l'anterior taula la diferència de pes.

TAULA 1.9. Comparativa de formats per a fotografia en alta resolució

Imatge	Mida	Format	Compressió	Pes
Skate	1.920x1.080	PNG		1.882 KB
Skate	1.920x1.080	JPEG	100%	1.386 KB
Skate	1.920x1.080	JPEG	50%	105 KB
Skate	1.920x1.080	JPEG	10%	43 KB
Skate	1.920x1.080	JPEG	0%	31 KB

Normalment, per a fotografies es fa servir el format **JPEG**, amb una qualitat al voltant del 60%; en molts casos pràcticament no es nota la diferència. En alguns casos la qualitat pot ser molt important, i en uns altres ho pot ser reduir la mida, llavors cal anar provant fins a arribar a un compromís entre qualitat i compressió.

Però, segons el format, el contingut de la imatge també és important: vegeu, en la taula 1.10, com canvia el pes si en lloc de fer servir una fotografia fem servir una imatge amb formes i colors simples:

TAULA 1.10. Comparativa de formats per a fotografia en alta resolució

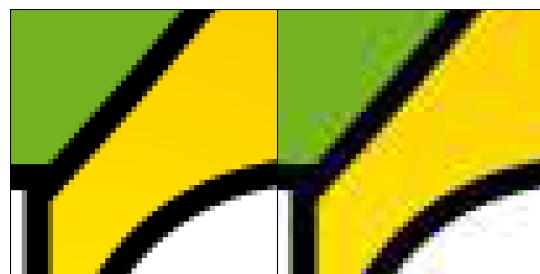
Imatge	Mida	Format	Compressió	Pes
Forma simple	1.920x1.080	PNG		21 KB
Forma simple	1.920x1.080	JPEG	100%	63 KB
Forma simple	1.920x1.080	JPEG	50%	34 KB
Forma simple	1.920x1.080	JPEG	10%	29 KB
Forma simple	1.920x1.080	JPEG	0%	27 KB

Full HD 1080p

Amb Full HD ens referim a una pantalla d'alta definició que suporta 1080 p d'alçada. Es tracta d'una resolució de 1.920x1.080 píxels, amb una relació d'aspecte de 16:9.

A banda del color, també es nota la diferència entre **formats**, perquè en aquest cas, amb el format JPEG es generen artefactes que distorsionen la imatge quan només hauria d'haver-hi colors plans com es pot apreciar a la figura 1.15.

FIGURA 1.15. Comparativa del detall entre una imatge PNG i una imatge JPEG



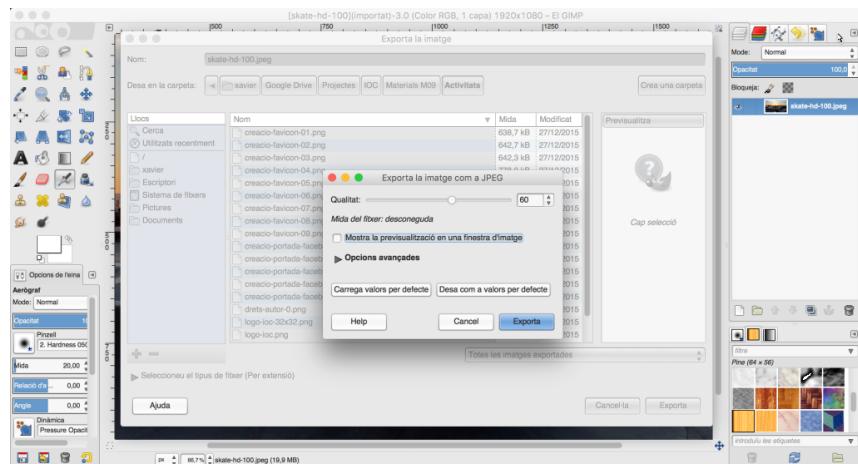
A l'esquerra, la imatge ampliada en format PNG; a la dreta, la mateixa imatge ampliada en format JPEG amb un 50% de compressió.

Com exportar una imatge amb diferent format a GIMP

Encara que no tots els programaris d'edició gràfica funcionen de la mateixa manera, habitualment l'exportació d'imatges és força similar. A continuació podeu veure els passos a seguir per exportar una imatge amb format JPEG i una qualitat del 60%:

1. Obriu la imatge que voleu optimitzar amb GIMP: seleccioneu l'opció del menú *Fitxer / Obre...* i seleccioneu el fitxer al vostre equip.
2. Seleccioneu l'opció del menú *Fitxer / Exporta com a...*
3. S'obrirà la finestra *Exporta la imatge*, on podeu seleccionar el directori on guardar-la i el nom del fitxer; a la part inferior, desplegueu les opcions de *Seleccioneu el tipus de fitxer* i cliqueu sobre *Imatge JPEG* (o canviieu l'extensió del nom del fitxer per .jpg).
4. Cliqueu al botó *Exporta*.
5. Veureu una finestra com la de la figura 1.16 anomenada *Exporta la imatge com a JPEG*.
6. Canvieu la qualitat al 60% i cliqueu al botó *Exporta*.

FIGURA 1.16. Exportació d'una imatge amb format JPEG a GIMP



Podreu trobar la vostra imatge amb format JPEG i qualitat 60% a la carpeta que hagiu especificat.

Càrrega de diferents imatges per a diferents mides

En molts casos, quan tractem amb imatges decoratives que formen part de la web i no dels continguts, en lloc de fer servir les etiquetes d'imatge podem establir aquestes imatges com a `background` d'un element (per exemple un `div`), de manera que fent servir Media Queries podem canviar aquesta imatge per una d'una resolució apropiada per al dispositiu en el qual s'està navegant per la pàgina.

Per exemple, si en una pàgina web tenim com a fons un dibuix que ocupa tot l'ample de la pantalla, no té gaire sentit que en navegar per aquesta pàgina amb un mòbil amb una resolució de 480x320 píxels descarreguem una imatge que fa 1.920x1.080 píxels.

Lús de Media Queries també és un element clau del *responsive design* (o disseny web adaptatiu), del qual podeu trobar més informació en aquest enllaç: goo.gl/pKZdrB.

Una altra tècnica per augmentar la velocitat de descàrrega és reduir el nombre de peticions; per exemple, si la nostra pàgina fa servir moltes icones i fem servir una imatge per a cadascuna d'aquestes, el nombre de peticions que s'ha de fer al servidor augmenta ràpidament.

Per evitar aquest problema hi ha dues solucions:

- Enviar tota la informació d'aquestes imatges en un sol fitxer (**molt poc popular**).
- Fer servir una font especial que en lloc de lletres contingui aquestes imatges (**això és el que s'usa més habitualment**).

Tots dos sistemes s'apliquen d'una manera similar; quan fem servir un únic fitxer d'imatges li diem **spritesheet** o **atles**.

La idea és que coneixent en quines coordenades comença cada imatge i la seva mida podem crear classes CSS que facin servir aquesta informació, de manera que en crear un element d'aquest tipus (per exemple, amb les etiquetes `` o `<i>`) s'insereixi la imatge retallada al nostre document.

```

1 <style>
2   ul {
3     list-style:none;
4     padding: 0;
5   }
6
7   .flag
8   {
9     display: inline-block;
10    width: 16px;
11    height: 11px;
12    line-height: 11px;
13    background-image: url("//m09-u2.surge.sh/imatges/flags.png");
14    background-position: 0 0;
15    background-repeat: no-repeat;
16  }
17
18  .flag-es { background-position: 0px 0px; width: 16px; height: 11px; }
19  .flag-catalonia{ background-position: -16px 0px; width: 16px; height: 11px; }
20  .flag-fr { background-position: -32px 0px; width: 16px; height: 11px; }
21</style>
22
23 Imatge original: <img src = "//m09-u2.surge.sh/imatges/flags.png" alt="banderes
24           " />
<ul>
```

```

25  <li><i class="flag flag-es"></i> Espanya</li>
26  <li><i class="flag flag-catalonia"></i> Catalunya</li>
27  <li><i class="flag flag-fr"></i> França</li>
28  </ul>

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/zqLMaz, i el resultat a la figura 1.17.

FIGURA 1.17. Exemple d'utilització d'un atles per mostrar banderes

Imatge original: 

 Espanya
 Catalunya
 França

Primerament, s'ha declarat una classe `flag` amb la mida de les banderes (16 x 11 píxels) i a continuació una classe particular per a cadascuna, de manera que desplacen la imatge com de fons:

- La bandera d'Espanya no es desplaça perquè es troba en primera posició.
- La bandera de Catalunya es desplaça 16 píxels (just l'amplada de la imatge).
- La bandera de França es desplaça 32 píxels (l'amplada de les dues imatges anteriors).

Per utilitzar les noves banderes només cal que afegiu la classe general i l'específica a un element buit i es representarà en aquesta posició; per exemple: `<i class="flag flag-catalonia"></i>`.

Però en el cas d'haver de treballar amb moltes imatges, o que la mida d'aquestes no sigui igual, fer els càlculs de les coordenades no és trivial. Per aquest motiu, el seu ús no està gaire estès, encara que quan es desenvolupen jocs amb HTML5 aquesta és la manera més habitual de trobar les animacions o imatges, ja sigui en *spritesheets* per a un element individual amb les seves animacions, o un atles complet amb totes les imatges del joc.

Tot i així, si la velocitat és un factor crític per al vostre projecte, aquesta és una opció a tenir en compte si teniu més d'un parell d'imatges decoratives que feu servir en la majoria de les pàgines.

En canvi, l'ús de fonts és molt popular, ja que existeixen llibreries molt fàcils de carregar i que només requereixen consultar la pàgina per veure quin nom rep la classe que ens interessa.

Per exemple, **Font Awesome** (vegeu la figura 1.18) i **Bootstrap Glyphicons**, que en lloc de carregar un atles el que fan és carregar una font, i fent servir diferents classes CSS s'afegeixen aquestes icones.

Més informació sobre atles i 'spritesheets'

Podeu trobar més informació en el següent enllaç: goo.gl/fMMys0, i com fer-les servir en jocs en aquest altre: goo.gl/V10Bx8.

Glyphicon és un component que forma part de la biblioteca Bootstrap.

FIGURA 1.18. Mostra d'ícones Font Awesome

Font Awesome inclou una gran varietat d'ícones de marques molt conegudes, generalment relacionades amb Internet.

La principal diferència entre tots dos és que Font Awesome incorpora moltes més ícones i són més específiques, incloent-hi ícones de companyies populars com Facebook, Twitter, Vimeo, Youtube, Chrome, etc. i ícones emblemàtiques com els botons de “M’agrada”, “Retweet”, etc.

Per altra banda, Glyphicons, en contenir menys imatges, és més lleugera; és qüestió de determinar en quina biblioteca es troben les ícones que necessiteu i fer servir una o una altra segons les vostres necessitats.

No oblideu que, en tractar-se de fonts, aquestes imatges poden **ajustar-se a la mida** que desitgeu sense perdre qualitat, ja que es tracten com a imatges vectorials.

En conclusió, a l'hora d'optimitzar un recurs gràfic les principals opcions són:

- Reduir la mida de la imatge al mínim necessari, de manera que s'ajusti a les vostres necessitats.
- Si es tracta d'una fotografia o una imatge amb molta varietat de formes i colors, exportar-la com a JPEG amb un nivell de compressió adequat, generalment al 60%.
- En el cas de les imatges vectorials o de colors plans i formes senzilles, si no voleu que es generin artefactes podeu exportar-la en el mateix format (GIF o PNG), però reduint el nombre de colors.
- Si la imatge a utilitzar s'empra com una icona o símbol, considereu reemplaçar-la per un dels símbols de FontAwesome o GlyphIcons, específicament si ja teniu carregades aquestes biblioteques.

- Si heu de fer servir moltes icones o símbols propis podeu crear un atles amb totes les imatges i fer servir CSS per inserir-les.

1.2 Elements d'àudio i vídeo per al web

Avui dia és molt fàcil obtenir recursos de vídeo i àudio original, ja que gràcies als dispositius mòbils quasi tothom té accés a micròfons i càmeres de vídeo integrats. Però encara que la qualitat d'aquests pugui ser molt bona, és poc probable que es puguin utilitzar sense fer cap tipus d'edició ja sigui per eliminar algunes parts, per afegir efectes o per substituir l'àudio d'un vídeo per exemple, cosa que obliga a fer servir un programari específic d'edició.

Però la creació de continguts audiovisuals no es limita només a la creació de vídeos. Des de principis dels anys 90 es poden trobar a Internet petites animacions incrustades als llocs web gràcies al format GIF89a i actualment, a través de CSS, és possible crear tot tipus d'animacions a partir dels elements HTML, imatges, combinacions de colors, textos, canvis d'opacitat, transicions, etc.

1.2.1 Programari per manipular i editar àudio i vídeo

Molt sovint, quan s'ha de portar a terme algun tractament d'àudio o vídeo, aquest és proporcionat per la empresa o client i el que s'haurà de fer serà optimitzar-lo per fer-lo servir en el medi que us interessi: exportar-lo a diferents formats, optimitzar la seva compressió, retallar un tros de vídeo, etc.

Àudio

Encara que hi ha molts programes professionals per a l'edició d'àudio, no hi ha cap que destaquí en particular pel tractament de so per al web. En general, les accions que es necessita portar a terme són:

- Exportar so en diferents formats.
- Modificar el volum.
- Afegir efectes especials de so (eco, modificar el to, etc.).
- Retallar i concatenar pistes d'àudio.

Jocs independents ('indie')

Es tracta de jocs desenvolupats per una sola persona o petits equips de desenvolupadors que no compten amb el suport d'un distribuïdor de jocs. Habitualment, la distribució es fa a través d'internet, fent servir plataformes digitals: Steam, App Store, Google Play, Chrome Web Store, etc.

En aquests materials ens centrarem en les característiques d'Audacity i el farem servir per als exemples. Es tracta d'un programari ben valorat per fer tant l'edició d'elements multimèdia senzills com per a l'edició d'efectes d'àudio per a jocs independents.

Audacity és un programari lliure multiplataforma, disponible per a Windows, macOS i Linux, que es pot descarregar des del següent enllaç: audacity.es.

Entre les opcions que ens ofereix trobem:

- Importar en diferents formats.
- Exportar en els formats més habituals: MP3 i OGG, entre d'altres.
- Mesclar dos o més sons diferents a la mateixa o en diferents pistes.
- Retallar els sons.
- Reduir el soroll.
- Aplicar efectes.



Popularitat d'Audacity

Com es pot veure a alternativeto.net/tag/audio-editor, un lloc especialitzat en alternatives de programari, a la secció d'edició d'àudio trobem que Audacity és el preferit amb diferència, superant en més de 15 vegades la primera opció de programari privatiu.

Vídeo

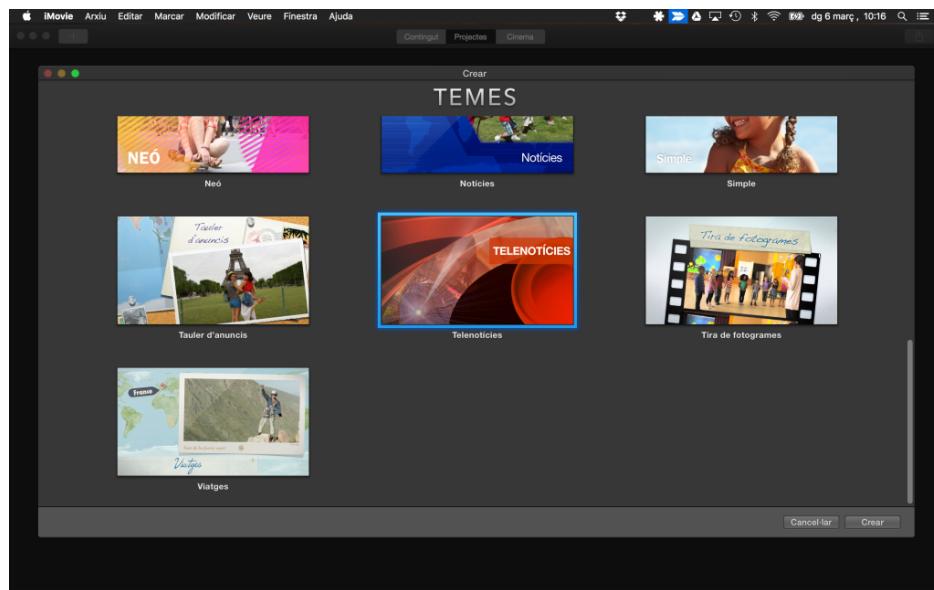
Quan parlem de vídeo cal distingir entre el programari que permet capturar-lo i el que permet editar-lo i exportar-lo a diferents formats.

Entre les opcions de programari gratuït podeu trobar les següents:

Instal·lació de Windows Movie Maker

Tot i que ha estat discontinuat, l'última versió és compatible amb Windows 10 i es pot descarregar del següent enllaç: goo.gl/qMXp7m.

- **Windows Movie Maker** (Windows): aquest programari es podia trobar en les instal·lacions del sistema operatiu Windows en versions anteriors, però no ha estat actualitzat des del 2012 i no s'inclou en el sistema operatiu a partir de la versió Windows 10. Permet enllaçar vídeos d'una manera molt senzilla i no compta amb gaires opcions, raó per la qual és molt fàcil de fer servir per als principiants. Només està disponible per a usuaris de Windows.
- **iMovie** (macOS): aquesta és l'opció més completa, ja que és intuïtiu i ofereix una gran col·lecció de sons, animacions i efectes per afegir als nostres vídeos (vegeu la figura 1.19). A més, ofereix també opcions avançades com per exemple exportar vídeos en 4K, fet que el converteix en la més completa de les tres opcions encara que limitat a usuaris de macOS.

FIGURA 1.19. Pantalla de creació d'una nova pel·lícula d'iMovie

- **Avidemux** (Windows, macOS, Linux): aquesta opció és molt més limitada i més complicada de fer servir que els dos anteriors, però l'avantatge que té és que es pot emprar en qualsevol plataforma i que permet la creació de *scripts* fent servir ECMAScript i la creació de treballs per lots el que permet, per exemple, canviar de format tots els fitxers d'un directori o aplicar-los un mateix filtre.

Normalment, si les necessitats d'edició són molt bàsiques, optareu per fer servir el programari més adequat al vostre sistema operatiu, Windows Movie Maker per a Windows i iMovie per a macOS, emprant Avidemux només si sou usuaris de Linux o necessiteu processar molts fitxers alhora (per exemple, per canviar-ne el format).

Entre les opcions de programari privatius totes són molts similars, encara que cal destacar que és difícil trobar alguna que funcioni nadiuament sota Linux, i el factor determinant per escollir una o una altra serà el sistema operatiu que fem servir i el cost del programari.

Al contrari que en el cas del programari gratuït, aquests programes són força complexos i requereixen dedicar algun temps per aprendre el seu funcionament abans de treure'n profit:

- **Adobe Premiere CC** (Windows, macOS): www.adobe.com/es/products/premiere.html
- **Sony Vegas Pro** (Windows): www.sonycreativesoftware.com/es/download/trials/vegaspro
- **Final Cut Pro** (macOS): www.apple.com/es/final-cut-pro
- **Pinnacle Studio Ultimate** (Windows): www.pinnaclesys.com/publicsite/sp/products/studio/ultimate

'Screencasting'

És una tècnica que consisteix en la captura de l'entrada o sortida d'un o més dispositius connectats a un ordinador, per exemple d'una *webcam* i/o una o més pantalles.

Un dels usos més habituals per als *screencasts* és la creació de tutorials, cursos i demostracions gravades en temps real.

Entre les opcions de *software* gratuït només n'hi ha una que destaca sobre la resta, i ho fa amb força diferència, Open Broadcaster Software, per a Windows, macOS i Linux). Aquest programari ens permet gravar captures de pantalla completa, *webcam*, finestres d'aplicacions o retransmetre en *streaming* a través de plataformes com Twitch o YouTube. No ofereix les possibilitats de muntatge i postproducció que ofereixen les alternatives de pagament, però és possible fer la gravació simultània de diferents fonts d'entrada, per exemple de la pantalla i la *webcam*.

En l'àmbit del *software* privatiu trobem dues opcions molt esteses. Totes dues tenen un preu similar i ofereixen pràcticament les mateixes prestacions, encara que Screen Flow només està disponible per a macOS. Són les següents:

- Camtasia (Windows, macOS): www.techsmith.com/camtasia.html
- Screen Flow (macOS): www.telestream.net/screenflow/overview.htm

Tots dos són força senzills de fer servir i ofereixen opcions d'edició en el mateix paquet:

- Edició de múltiples pistes de vídeo i àudio.
- Afegir anotacions i animacions.
- Composició dels vídeos (per exemple, canviar la posició del vídeo de la *webcam* sobre el de la finestra).
- Publicar directament a plataformes de vídeo com YouTube i Vimeo.

A continuació trobareu un exemple fent servir Open Broadcaster Software, perquè en ser multiplataforma es pot seguir amb Windows, macOS i Linux.

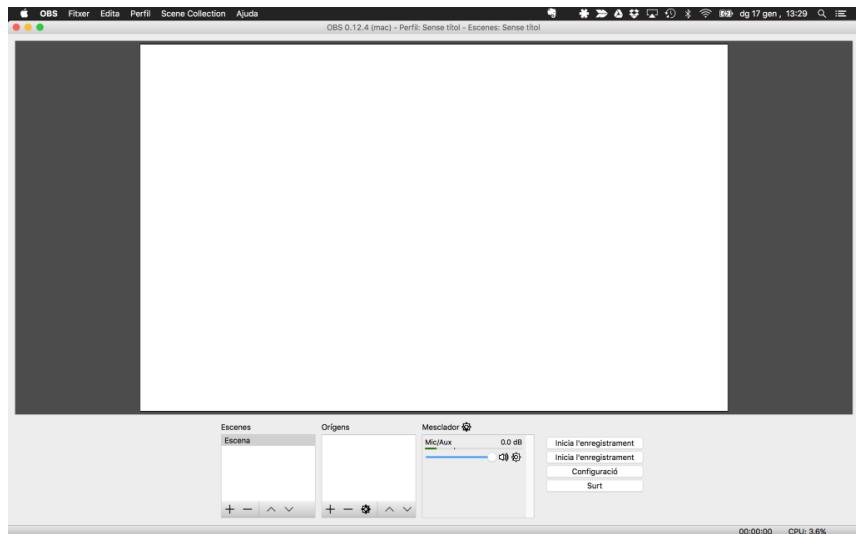
Captura de vídeo amb Open Broadcaster Software

El primer que heu de fer és descarregar-lo de la següent adreça: obsproject.com/download. Una vegada descarregat, s'ha d'instal·lar. Els passos són diferents en cada plataforma, però no cal configurar res.

Obriu l'Open Broadcaster Software (abreviat com OBS). Veureu un bloc central en negre i a sota quatre seccions, la primera anomenada *Escenes*, la segona *Orígens*, la tercera *Mesclador* i a la quarta una sèrie de botons com es pot apreciar a la figura 1.20.

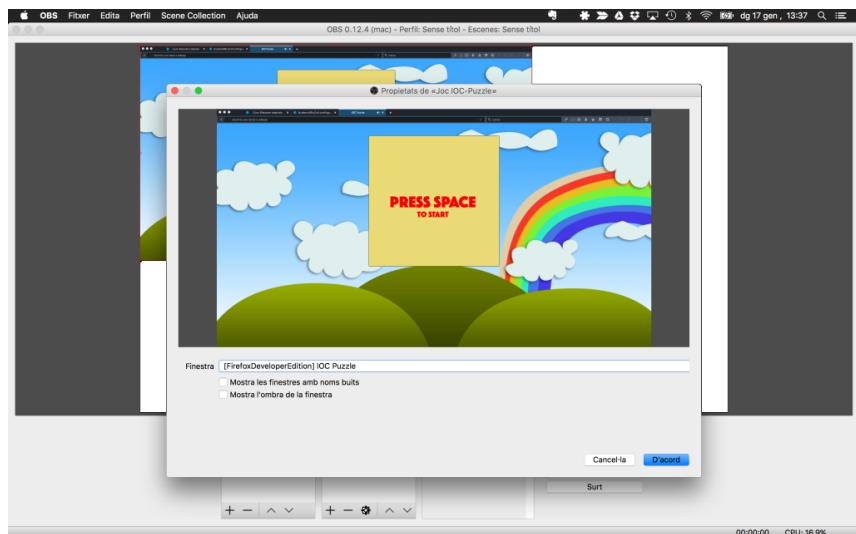
FPS ('frames per second')

Els *frames* per segon representen el nombre d'imatges que es mostren en un segon. Els formats de vídeo per a televisió analògica com PAL i NTSC mostren 24 i 29 FPS, respectivament, mentre que els videojocs han de córrer a 60 FPS.

FIGURA 1.20. Pantalla d'inici d'Open Broadcaster Software

Cliqueu al símbol "+", a sota d'*Orígens*, i seleccioneu *Captura de finestra*. Aquí podeu fer servir el nom que vulgueu.

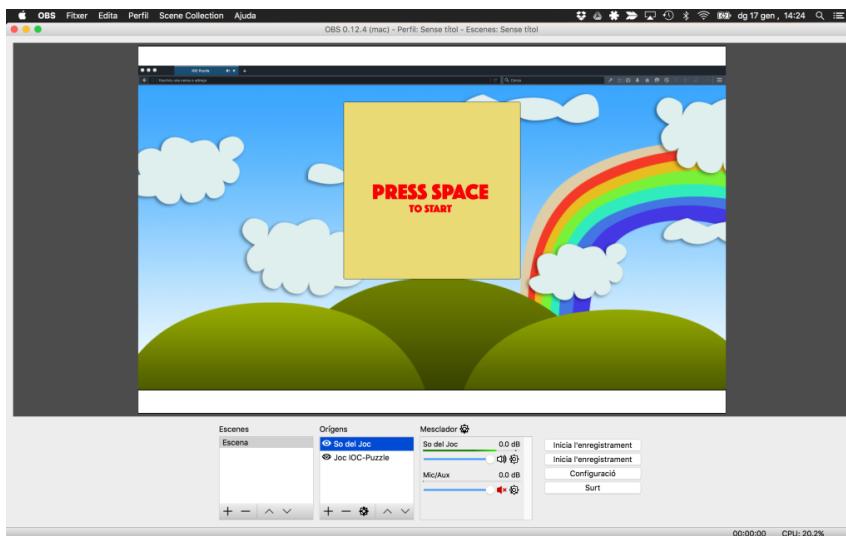
A continuació podreu seleccionar quina finestra voleu capturar; en aquest exemple capturareu la finestra on s'està executant el contingut que vulgueu enregistrar, com es pot apreciar a la figura 1.21.

FIGURA 1.21. Selecció de finestra com a origen

Com que interessa incloure el so de l'aplicació, afegireu la *sortida d'àudio*. Per defecte està habilitada l'entrada del micròfon si en teniu un connectat; en aquest cas, com que no interessa gravar l'àudio del micròfon, heu de fer clic sobre la icona que el mostra per desconnectar-lo (es posarà de color vermell).

Una vegada afegits els canals de vídeo i àudio només queda clicar sobre el botó *Inicia l'enregistrament*. Si us fixeu en la figura 1.22, aquesta versió localitzada al català mostra dos botons amb el mateix text. El botó superior és per iniciar la retransmissió en *streaming* si ho heu configurat; l'inferior és el que iniciària la gravació. Una vegada el cliqueu començarà la reproducció fins que el torneu a clicar per aturar-lo.

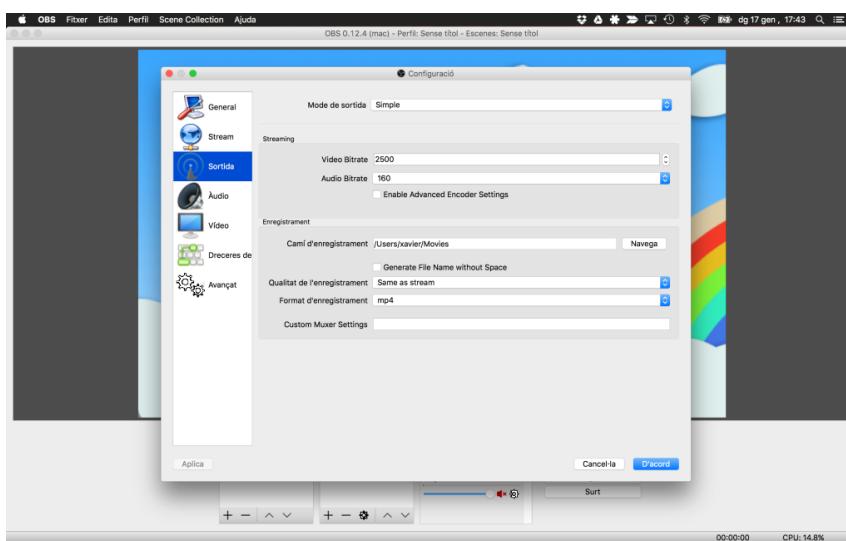
En clicar sobre la captura de vídeo les vores es posen de color vermell. Això indica que està seleccionada i que és possible canviar la mida i moure-la sobre el fons negre; aquest mateix sistema es fa servir per reorganitzar les diferents àrees de captura de vídeo diferents (per exemple, pantalla i *webcam* alhora).

FIGURA 1.22. OBS amb un origen de vídeo i d'àudio configurat

S'ha de tenir en compte que aquest fons negre serà visible en el vídeo, així que o bé s'ha d'ajustar la mida del vídeo a la que heu de fer servir, o bé s'han d'ajustar els continguts per cobrir-ho tot. Aquesta és l'opció que es farà servir en aquest exemple, de manera que a més a més s'oculta la interefície del navegador.

Amb això ja tindreu el vostre primer vídeo creat, ara només caldrà obrir la carpeta on l'hàgiu guardat i obrir-lo amb el vostre reproductor multimèdia favorit per comprovar-ne el resultat.

Com que d'entrada OBS no us demana cap opció de configuració, el format i la localització del vídeo segurament no són els que desitgeu. Per canviar aquestes opcions, **abans de començar a gravar**, haureu de fer clic sobre el botó *Configuració* i fer els canvis necessaris a la secció de sortida, com es pot veure a la figura 1.23:

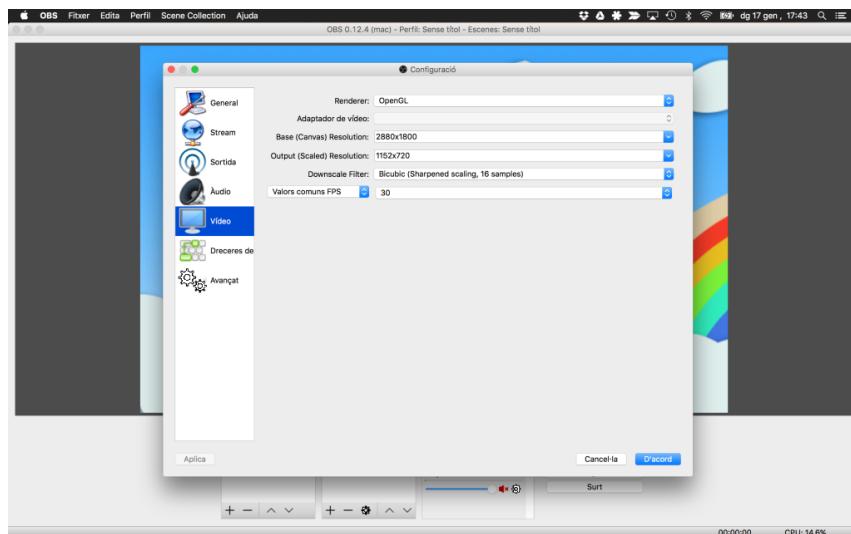
FIGURA 1.23. Opcions de configuració de sortida

- **Bitrate:** a major *bitrate*, més ocupa el vídeo, però s'obté una major qualitat, tant per al vídeo com per a l'àudio.
- **Camí d'enregistrament:** és on es guarda el fitxer generat.
- **Format d'enregistrament:** és el format en el qual es guarda el vídeo.

I a la secció de vídeo podeu configurar el següent (vegeu la figura 1.24):

- **Base (Canvas) Resolution:** la resolució d'entrada; en cas de tenir múltiples pantalles, es mostraran aquí les possibilitats.
- **Output (Scaled) Resolution:** la resolució del llenç negre sobre el qual col·loquem el nostre vídeo. S'ha de tenir en compte que aquestes són proporcionals a l'anterior, deixant sempre fixa l'alçada per corresponder amb els tipus més habituals (1080 p, 720 p, etc.). Per exemple, si la vostra pantalla té com a resolució activa 1920x1080, es mostrarà l'opció 1920x1080 i 1280x720, però si la resolució activa és 2880x1800, les corresponents seran 1728x1080 i 1152x720, respectivament.
- **Valors comuns FPS:** nombre de *frames* per segon. Normalment, 30 o 60 FPS si volem una sensació més fluida. S'ha de tenir en compte que si fem servir un nombre de *frames* major es reduirà el rendiment del sistema i augmentarà l'espai ocupat pel vídeo.

FIGURA 1.24. Opcions de configuració de vídeo



S'ha de tenir en compte que OBS no permet guardar projectes. Al contrari que altres programes d'edició, OBS no fa servir un sistema de projectes que permeti desar les dades d'un projecte concret; en canvi, es guarden aquestes configuracions al quadre anomenat *Escenes* i cada vegada que obrim el programa podem seleccionar quina escena aplicar.

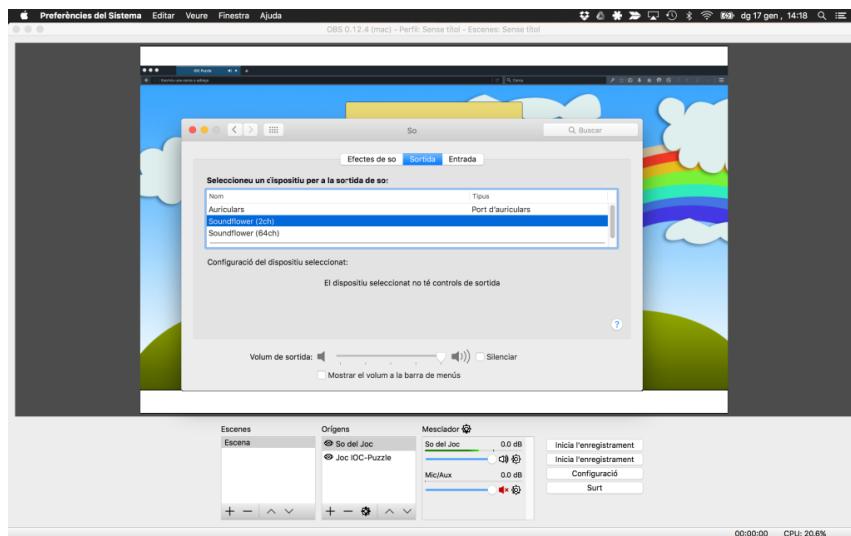
Finalment, podeu accedir a la carpeta de vídeos desats directament des de l'opció *Fitxer / Mostra els enregistraments*.

A macOS no es pot capturar el so de sortida directament, i és necessari descarregar una extensió que afegeix aquesta opció al sistema. Una de les opcions és fer servir Soundflower, que es pot descarregar en aquest enllaç: goo.gl/U4XSUt.

Una vegada instal·lat, en seleccionar captura l'àudio d'entrada veureu tres opcions: per defecte, Soundflower 2ch i Soundflower 64ch.

La versió disponible del controlador en el moment de la redacció d'aquests materials no disposa d'un control que es pugui afegir a la barra superior per seleccionar el canal de sortida, així que heu d'anar a *Preferències / Àudio* i a la pestanya de sortida marcar el canal pel qual vulgueu que es reproduexi el so, per exemple CH2, i aquest mateix és el que heu de seleccionar dins d'*OBS* per fer que la captura de so funcioni correctament, com es pot apreciar a la figura 1.25.

FIGURA 1.25. Finestra amb les preferències d'àudio de macOS.



Si canviem aquesta opció deixarem d'escoltar l'àudio pels canals habituals; en aquest cas, la sortida d'auriculars.

Cal tenir en compte que ja no se sentirà el so pel canal normal, i si voleu tornar a sentir-lo pels auriculars o els altaveus haureu de tornar a seleccionar el dispositiu de sortida original.

La **configuració** és global, s'aplica a totes les escenes, de manera que si canviieu la resolució afecta totes i no tan sols la que tingueu seleccionada.

1.2.2 Àudio: formats i conversions de formats (exportar i importar)

Un dels majors problemes que es troben a l'hora de treballar amb àudio amb HTML5 és que no tots els navegadors suporten els mateixos formats, ja que l'especificació del llenguatge no obliga a implementar un o un altre. Això obliga els desenvolupadors a implementar a les pàgines diferents alternatives per assegurar que els usuaris de la pàgina podran reproduir l'àudio.

Encara que hi ha algunes variacions, els tres tipus més freqüents són:

- **PCM:** format d'àudio sense comprimir, és un format sense pèrdua. Normalment, les gravacions de so es fan en aquest format i després s'exporten a un format comprimit per optimitzar l'espai.
- **MP3:** aquest és un format basat en la compressió amb pèrdua. És molt utilitzat en diversos àmbits, però fins fa poc no era suportat per Firefox per ser un format patentat i Mozilla volia evitar problemes de patents.
- **Vorbis Ogg:** aquest també és un format basat amb la compressió amb pèrdua, però en aquest cas és un format lliure de patents, per això va ser l'opció escollida per l'equip de Firefox.

Podeu trobar més informació sobre formats multimèdia en HTML5 en aquest enllaç: goo.gl/H9q3ER.

Dit això, queda clar que els dos candidats possibles per incloure a la web són els formats MP3 o OGG. A la taula 1.11 podeu veure la compatibilitat de formats per a diferents navegadors:

TAULA 1.11. Compatibilitat de formats d'àudio

Format	Extensió	Chrome	Firefox	Internet Explorer	Opera	Safari
PCM	WAV	Sí	3.5	No	10.50	3.1
OGG Vorbis	OGG	Sí	3.5	No	10.50	No
MP3	MP3	Sí	Sí	9.0	Sí	3.1

El nombre correspon a la primera versió en què està disponible.

Encara que pel nom es podria pensar que **MP4** és una versió més actualitzada del format MP3, no és així. El format MP4 és un format de contingut multimèdia que pot emmagatzemar àudio, vídeo, subtítols i imatges, entre altres tipus d'elements. Habitualment es trobaran fitxers amb format MP4 com a continguts de vídeo; per aquesta raó, en aquests materials quan es menciona aquest format s'ha d'interpretar que es tracta d'un contingut de vídeo.

Actualment és prou segur fer servir el format MP3, però si voleu assegurar que el vostre àudio es reproduirà en tots els navegadors (que suporten HTML5) només heu d'afegir el mateix fitxer d'àudio en format OGG i MP3 i incloure'ls tots dos dins de l'element `audio` en el bloc HTML fent servir l'element `source`.

Ús de les etiquetes d'àudio i vídeo

Podeu trobar més informació sobre com fer servir aquests elements en el següent enllaç: goo.gl/XB1rQ4.

```

1 <audio controls>
2   <source src="http://m09-u2.surge.sh/so/Batty_McFaddin.mp3" type="audio/mp3">
3   <source src="http://m09-u2.surge.sh/so/Batty_McFaddin.ogg" type="audio/ogg">
4   Ho sento, el teu navegador no suporta audio d'HTML5
5 </audio>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/XKpqbj.

Com podeu veure a l'exemple, es troba primer com a font el fitxer MP3 i si aquest es pot reproduir, s'aturarà l'execució del codi. En cas contrari, es provarà amb el fitxer OGG i si aquest format tampoc es reconeix, es mostrarà el missatge.

Opcions de l'element "audio"

Moltes de les característiques de l'element `audio` només són accessibles des de JavaScript, però hi ha unes quantes opcions que es poden activar directament des del codi HTML. Podeu trobar algunes d'aquestes opcions a la taula 1.12.

TAULA 1.12. Atributs de l'etiqueta "audio"

Atribut	Efecte
Autoplay	El so es reproduirà tan aviat com sigui possible.
Controls	Es mostraran els controls de reproducció.
Loop	El so és reproduirà des del principi en acabar.
Muted	El so s'iniciarà silenciat, s'haurà de canviar l'estat via codi.
SRC	Si només necessitem un format d'àudio podem fer servir aquest atribut en lloc de l'etiqueta <code>source</code> .

Es pot trobar més informació sobre els atributs d'àudio en el següent enllaç: goo.gl/fudzmr.

TAULA 1.12 (continuació)

Atribut	Efecte
Volume	Volum amb el qual es reproduirà l'àudio, entre 0.0 (silenci) i 1.0 (molt alt).

Hi ha més atributs, però no tenen cap efecte.

Per exemple, el següent codi mostraria els controls del reproductor, el fitxer es reproduiria immediatament i es repetiria contínuament.

```

1 <audio controls autoplay loop>
2   <source src="http://m09-u2.surge.sh/so/Batty_McFaddin.mp3" type="audio/mp3">
3   <source src="http://m09-u2.surge.sh/so/Batty_McFaddin.ogg" type="audio/ogg">
4   Ho sento, el teu navegador no suporta àudio d'HTML5
5 </audio>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/KVRzbb.



Element "audio" des de JavaScript

La inclusió de l'element `audio` a HTML5 permet anar més enllà i instanciar sons dinàmicament, ajustar el seu volum, reproduir i aturar la reproducció, etc. Això permet afegir, per exemple, sons que són reproduïts en fer clic en un botó, o afegir-los a jocs que s'executen al navegador.

A continuació veureu alguns exemples molt bàsics mostrant com es poden modificar les propietats d'aquests elements dinàmicament, i sense necessitar incloure'ls al marcantge HTML.

Tot i que l'element `audio` ja conté moltes funcionalitats, s'està treballant en una API més avançada, la **Web Audio Api**, que fa servir un sistema de connectors de manera que es poden fer múltiples modificacions i aplicar filtres al so abans d'utilitzar-lo.

Un exemple molt bàsic és el següent, la reproducció d'un so amb uns paràmetres concrets en carregar la pàgina:

```

1 <script>
2   var so = new Audio('audio.mp3');
3   so.volume = 0.5; // opcionalment podem establir el volum
4   so.loop = true; // indiquem que volem que es reproduexi indefinidament
5   so.play(); // reproduïm el so
6 </script>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/MKBNxj.

A continuació, podeu veure el mateix exemple però aquest cop podeu controlar la reproducció del so mitjançant un botó (funcionalitat que us permet crear els vostres propis reproductors d'àudio):

Web Audio Api

Aquesta API encara es troba en estat experimental i no és recomanable fer-la servir sense comprovar abans el nivell de suport en els navegadors que interessen. En podeu trobar més informació en el següent enllaç: goo.gl/tuYViq.

A causa de les limitacions actuals de **CodePen**, els exemples sobre àudio i vídeo no funcionen correctament perquè els fitxers d'àudio i vídeo no es troben a la plataforma. Per comprovar el seu funcionament heu de substituir el nom del fitxer per l'URL d'un fitxer d'àudio o vídeo apropiat.

```

1 <script>
2   var so = new Audio('audio.mp3');
3
4   function reproduir() {
5     so.play();
6   }
7 </script>
8
9 <button onclick="reproduir();">Reproducir</button>

```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/vLaoMQ.

Si proveu a fer clic repetidament sobre el botó, veureu que no té cap efecte. Fins que no acaba de reproduuir-se no torna a començar.

Quan es fan servir els elements d'audio des de JavaScript es disposa de més opcions, però hi ha una limitació molt important: no es pot reproduir el mateix element més d'una vegada simultàniament. Una solució a aquest problema és crear múltiples elements de so amb el mateix fitxer, de manera que cada vegada que fem un clic, l'element audio que es reproduceix és el següent disponible a la pila, que acostuma a ser cíclica, és a dir, en arribar a l'últim element es comença pel primer.

Ús de l'operació mòdul (%)

Quan es treballa amb iteracions cícliques, com és el cas del *pool*, que en arribar al final han de tornar al principi, es pot fer servir l'operació mòdul per calcular el valor corresponent a la posició en lloc de fer comprovacions i reinicialitzar el comptador de posició. Per exemple, en el nostre cas, com que treballem amb 10 valors tenim que: $5 \% 10 = \text{posició } 5$, $15 \% 10 = \text{posició } 5$.

Un exemple molt senzill per reutilitzar un *pool* de sons seria el següent:

```

1 <script>
2 var sons = [],
3     actual = 0;
4
5 // Creem 10 elements d'àudio i els afegim a l'array
6 for (var i = 0; i < 10; i++) {
7   sons.push(new Audio('audio.mp3'));
8 }
9
10 // En fer clic es reproduirà el so de l'índex de l'array actual i s'avançarà en
11 // un
12 function reproduir() {
13   sons[actual % 10].play();
14   actual++;
15 }
16 </script>
17 <button onclick="play();">Reproducir</button>

```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/WrKVQ.

Edició de fitxers d'àudio

Encara que al disseny de pàgines web no es fan servir gaires sons, pot ser que se'n necessiti algun d'específic per afegir a algun vídeo, per fer algun efecte d'algún joc en HTML5 o per a la creació d'un bàner publicitari.

Igual que passa amb les imatges, es poden trobar sons en formats sense pèrdua i amb pèrdua. Quan es parla d'un format PCM (Pulse Code Modulation) sempre es tracta d'un format sense pèrdua; d'aquest tipus es troben el format Waveform Audio File Format (WAVE) per a Windows i Audio Interchange File Format (AIFF) per a macOS.

Podeu trobar més informació sobre la modulació per impulsos codificats (PCM) a bit.ly/1gqjcgS.

En el cas de fer servir un format amb pèrdua de qualitat com OGG o MP3 necessitareu un **còdec (codificador/decodificador)** específic per tractar amb aquests formats. Encara que tots els reproductors inclouen alguns d'aquests còdecs i permeten reproduir aquests formats, és possible que per a alguns hagi de descarregar i instal·lar algun còdec específic. A la taula 1.13 podeu trobar informació sobre la compatibilitat de navegadors amb els formats més habituals.

TAULA 1.13. Compatibilitat de formats d'àudio

Format	Extensió	Chrome	Firefox	Internet Explorer	Opera	Safari
PCM	wav	Sí	3.5	No	10.50	3.1
OGG Vorbis	OGG	Sí	3.5	No	10.50	No
MP3	MP3	Sí	Sí	9.0	Sí	3.1

El nombre correspon a la primera versió en què està disponible.

Encara en reduir el nombre de bits i la freqüència la mida dels fitxers es redueix com es pot apreciar a la taula 1.14, també ho fa la qualitat fins a arribar a un punt en què només es pot apreciar soroll.

Per què 44100 Hz?

Amb 44100 Hz (44.1 kHz) es reproduïen totes les freqüències que és capaç de captar l'orella humana (són 22 kHz, però calen dues mostres).

TAULA 1.14. Comparativa del pes d'un so enregistrat amb diferents combinacions de bits, kHz, i tipus

bits	kHz	tipus	pes
32 bits	44 kHz	estèreo	~21.2 MB
24 bits	44 kHz	estèreo	~15.9 MB
16 bits	44 kHz	estèreo	~10.2 MB
16 bits	44 kHz	mono	~5.3 MB
16 bits	22 kHz	mono	~2.6 MB
16 bits	11 kHz	mono	~1.3 MB
8 bits	11 kHz	mono	~660 KB

El so comparat té una duració d'un minut.

La solució per reduir el pes és fer servir un format comprimit com OGG (vegeu la taula 1.15) o MP3 (vegeu la taula 1.16), que redueixen molt el pes del fitxer sense fer disminuir pràcticament la qualitat eliminant els sons que no són captats per l'orella humana.

TAULA 1.15. Comparativa d'un so en format OGG

Qualitat	Pes
10/10	~2.4 MB
5/10	~670 KB
0/10	~386 KB

Enregistrat en 32 bits, 44 kHz, estèreo, amb duració d'un minut.

TAULA 1.16. Comparativa d'un so en format MP3

Bitrate	Pes	Qualitat
320 kbps	~2.4 MB	MP3 alta definició
192 kbps	~1.4 MB	CD
96 kbps	~721 KB	Baixa qualitat

Enregistrat en 32 bits, 44 kHz, estèreo, amb duració d'un minut.

'Bitrate'

El *bitrate* són els bits per segon que ocupa un fitxer d'àudio en format MP3. Encara que el correcte és parlar de bit/s, l'abreviatura no estàndard bps o kbps és més popular.

Es pot descarregar Audacity en el següent enllaç:
www.audacityteam.org/download.

Còdec per exportar en MP3 amb Audacity

És possible que Audacity demani instal·lar un còdec per poder exportar en MP3, ja que en ser un format patentat no el poden incloure al seu programari. Si és necessari, us facilitarà un botó que en clicar-lo descarregará i instal·larà el còdec apropiat.

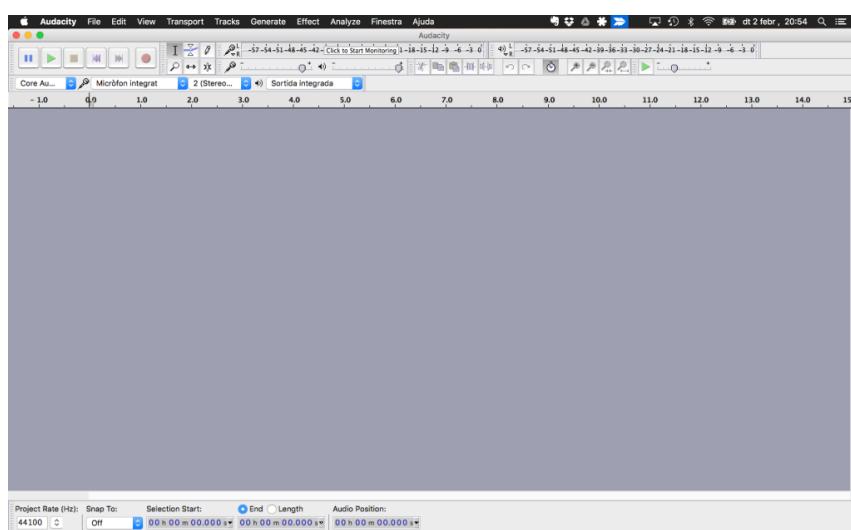
Amb un **nivell de compressió intermedi** s'aconsegueix una bona qualitat de so i un pes molt reduït. Els nivells més baixos no aconsegueixen reduint el pes en proporció, mentre que la qualitat es veu molt perjudicada.

Per determinar la qualitat quan s'exporta a format MP3, en lloc de fer servir un nombre de 0 a 10 com el format OGG, es fan servir els kilobits per segon (kbps). Aquest sistema permet tenir una idea més clara de les equivalències amb altres fonts d'àudio, com per exemple la qualitat de so d'un CD.

Gravar sons propis amb Audacity

Per seguir els passos d'aquest exemple de com es van gravar els sons del joc IOC Puzzle fent servir Audacity, cal tenir un micròfon connectat i correctament configurat.

Una vegada descarregat i instal·lat **Audacity** al vostre sistema, en obrir-lo trobareu la pantalla principal de l'aplicació (vegeu la figura 1.26). Aquí podreu veure l'entrada i sortida d'àudio, i els botons estàndards per gravar i reproduir.

FIGURA 1.26. Pantalla principal d'Audacity

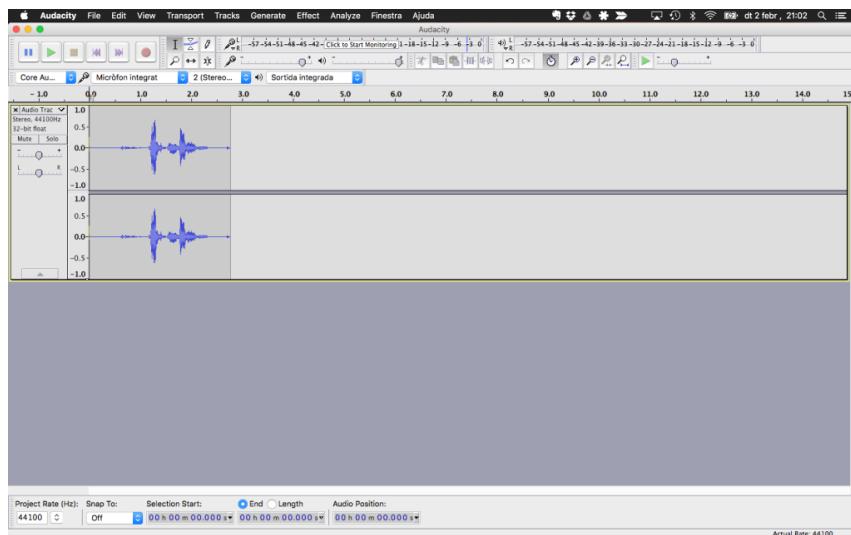
El primer pas és gravar un so, al qual després aplicareu diferents filtres. Heu de clicar en el botó *Record* (cercle vermell a la figura 1.27), fer el so o dir les paraules que vulgueu enregistrat, i clicar en el botó *Stop* (quadrat groc).

FIGURA 1.27. Botons de control



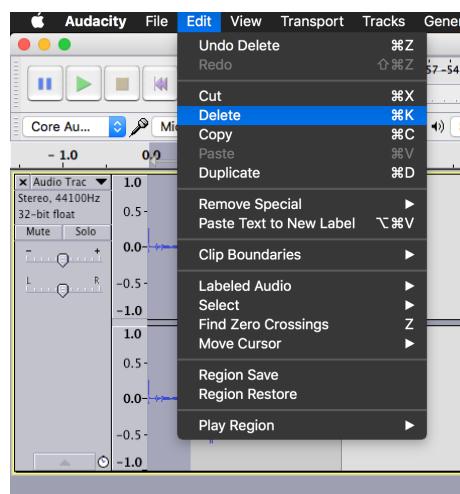
A mesura que s'enregistra el so veureu que ha aparegut una pista (*audio track*) amb la informació del so enregistrat, el senyal d'àudio generat pel so, el nombre de bits, els kHz, etc. Per defecte, el so s'enregistra en estèreo, per això es veuran dues ones a la pista (vegeu la figura 1.28).

FIGURA 1.28. Enregistrament de so



Si voleu eliminar part del so enregistrat, per exemple per tallar un tros al principi o al final, només heu de seleccionar quin fragment del senyal d'àudio voleu eliminar fent clic a un punt i arrossegant el cursor fins a seleccionar tot el que us interessi. A continuació, seleccioneu en el menú *Edit / Delete* (vegeu la figura 1.29).

FIGURA 1.29. Esborrar fragments d'àudio

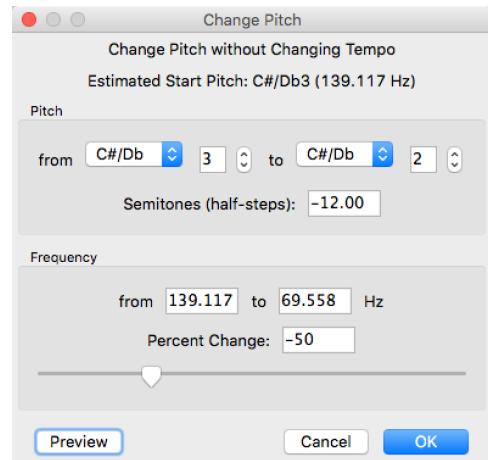


Audacity inclou una gran quantitat de filtres; a tall d'exemple, aplicareu *Change Pitch* per fer el so més greu, *Reverb* per afegir un efecte d'eco i *Normalize* per normalitzar el volum. Per

aplicar un filtre seleccioneu l'opció del menú *Effect* i el filtre desitjat, i si voleu escoltar com quedarà el so abans d'aplicar el filtre només cal clicar sobre el botó *Preview* de la finestra amb les opcions de l'efecte.

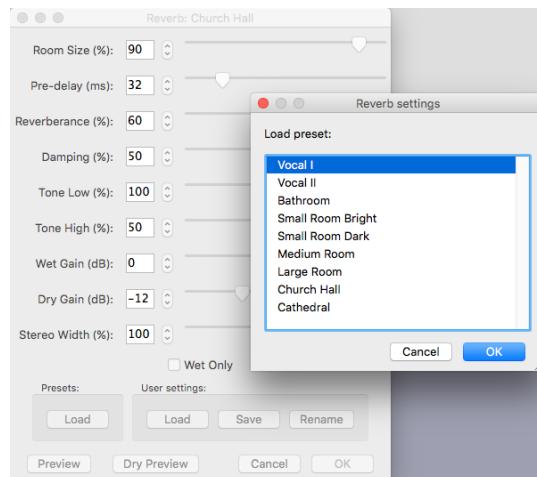
Comenceu amb *Effect / Change Pitch*, només cal canviar el *Percent Change* a -50% si voleu un so greu, o 75% si voleu que sigui agut; la resta de valors no cal modificar-los (vegeu la figura 1.30).

FIGURA 1.30. Opcions de l'efecte 'Change Pitch'



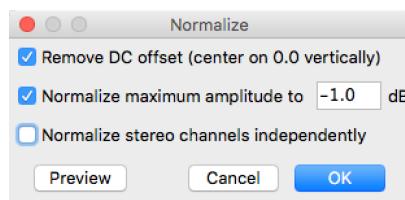
A continuació afegireu un efecte d'eco, seleccioneu *Effect / Reverb* i ajusteu els paràmetres al vostre gust. Cal destacar que aquest efecte compta amb una sèrie de valors predefinitos que es poden carregar fent clic sobre el botó *Load* (vegeu la figura 1.31).

FIGURA 1.31. Opcions de l'efecte 'Reverb'



Finalment, apliqueu *Effect / Normalize* (vegeu la figura 1.32).

FIGURA 1.32. Opcions de l'efecte 'Normalize'

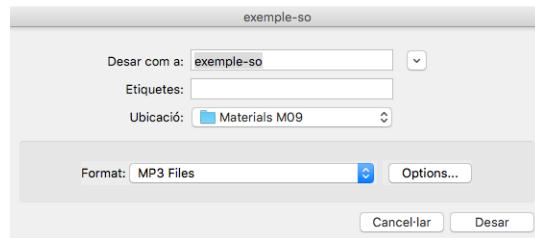


Deixeus les opcions per defecte i el so s'ajustarà entre 1.0 dB i -1.0 dB (que indica la potència del so). És important aplicar aquest filtre en últim lloc per assegurar que aquest serà el nivell de volum final del vostre so.

Si no ho heu fet abans deseu el fitxer amb el vostre projecte, seleccionant a l'opció del menú *File / Save project as...* triant el nom del fitxer i la ruta. Com indica el mateix programa, aquest no és el so per reproduir, sinó el projecte a partir del qual podreu continuar fent edicions més endavant.

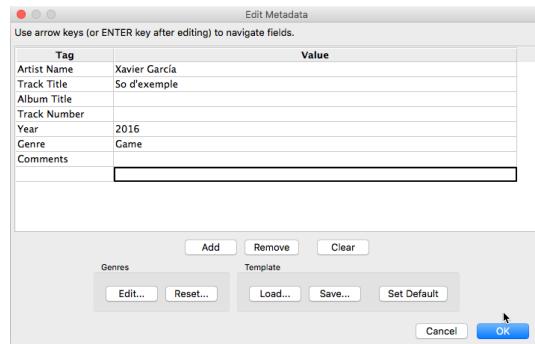
Per exportar el fitxer d'àudio creat i poder utilitzar-lo en altres aplicacions heu de seleccionar *File / Export Audio*. S'obrirà una nova finestra que permet seleccionar el nom del fitxer, la ruta i el format (i les seves opcions). Escolliu el format MP3 i a les opcions específiques de qualitat 192 kbps (vegeu la figura 1.33).

FIGURA 1.33. Exportació a MP3



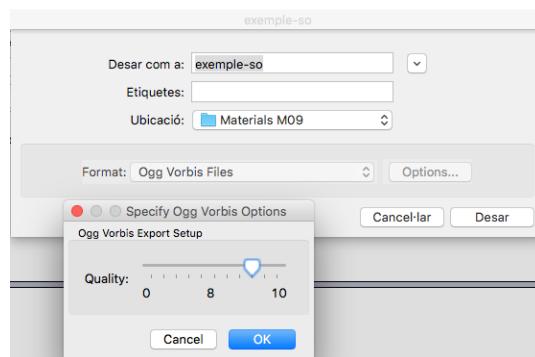
Podeu incloure les metadades que s'afegeiran al fitxer, que són opcionals (vegeu la figura 1.34).

FIGURA 1.34. Afegeir metadades del fitxer d'àudio



Per cobrir el màxim nombre de navegadors, exporteu-lo també en format OGG. D'aquesta manera, disposareu del mateix so en els dos formats suportats pels navegadors moderns. En aquest cas escolliu format OGG i qualitat 8 (vegeu la figura 1.35).

FIGURA 1.35. Exportació a OGG



Ja teniu el vostre so editat i disponible en format MP3 i OGG per incloure'l en els vostres continguts multimèdia.

Si el nombre d'efectes necessaris és molt gran, o són massa difícils de fer per vosaltres mateixos, sempre és possible recórrer a la cerca de recursos gratuïts o la compra d'aquests efectes en llocs especialitzats en la venda de sons i cançons *lliures de regalies*. S'ha de tenir molt de compte i comprovar abans de tot que us permet fer la llicència del recurs que adquiriu, ja que segons la utilització que li vulgueu donar podrà tenir un cost o un altre, o directament no ser permès.

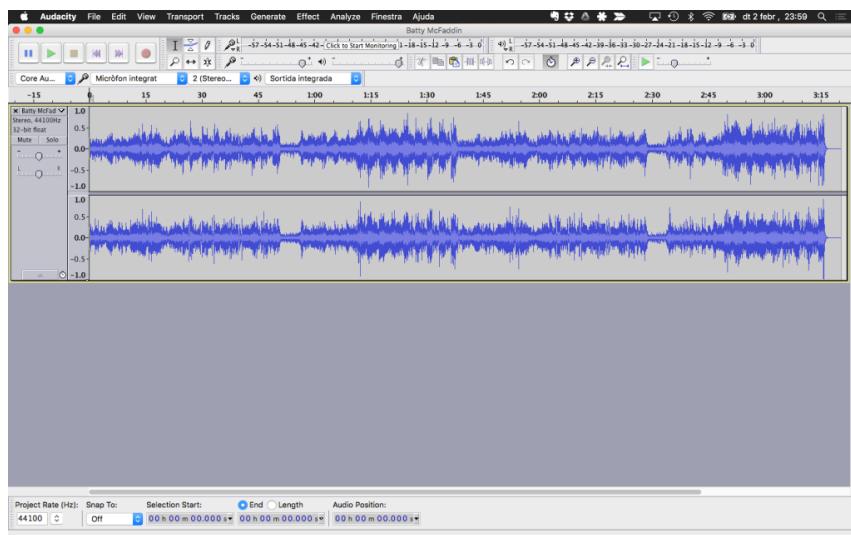
Molt sovint us trobareu que en haver d'afegir un fitxer d'àudio a una pàgina web o un joc només disposeu de l'àudio en un format i haureu d'exportar-lo a altres formats. Si aquest fitxer es troba en un format PCM el seu pes serà massa gran, així que s'haurà d'exportar a OGG i MP3 per assegurar la màxima compatibilitat possible. O potser el fitxer està enregistrat amb una qualitat massa gran i no s'ajusta a les vostres necessitats. Aquesta és una tasca molt simple, com podeu veure en el següent exemple.

Canvi de format i pes d'un fitxer d'àudio

El primer que necessitarreu serà el fitxer d'àudio per editar, en aquest cas descarregareu la cançó *Batty McFaddin* utilitzada al joc IOC Puzzle. Es pot trobar en el següent enllaç: goo.gl/0yDZOE.

Una vegada descarregat, obriu-lo a Audacity des de *File / Open*. Veureu tota la seva informació i el senyal d'àudio. Com podeu veure a la figura 1.36, es tracta d'un fitxer en estèreo, 32 bits i 44.100 Hz.

FIGURA 1.36. Fitxer d'àudio carregat



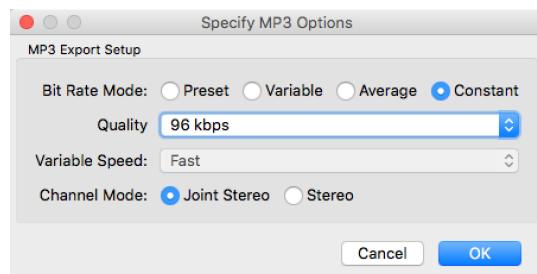
A la barra inferior podeu modificar els Hz que voleu per a aquest projecte; reduiu-los a 22.050 (vegeu la figura 1.37).

FIGURA 1.37. Hz del projecte



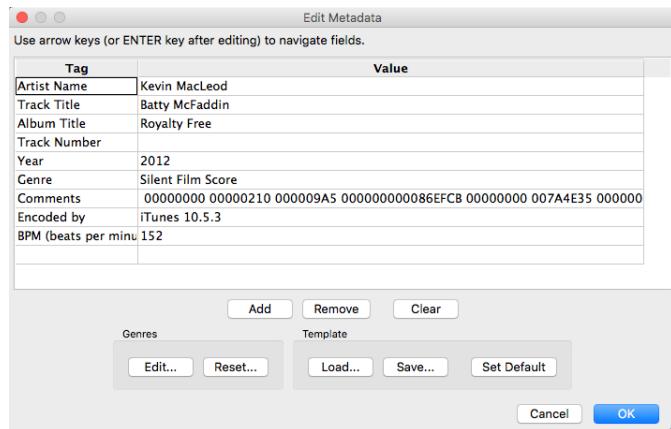
Exporteu-lo com a MP3 seleccionant *File / Export Audio*, i a opció seleccioneu 96 kbps com a qualitat (vegeu la figura 1.38).

FIGURA 1.38. Exportant àudio a MP3 amb qualitat reduïda



En aquest cas no modifiqueu les metadades, deixeu-ho tot com està i guardeu el fitxer (vegeu la figura 1.39).

FIGURA 1.39. Metadades de la cançó



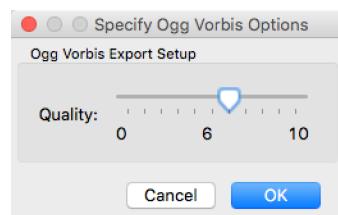
El fitxer ha passat d'ocupar 8 MB a ocupar-ne 2,4 a causa de la reducció de kbps i els Hz (vegeu la figura 1.40).

FIGURA 1.40. Comparació de pes entre el fitxer original i l'exportat



Repetiu el procés, aquest cop exportant-lo com a OGG (vegeu la figura 1.41). Torneu a *File / Export Audio* i a les opcions seleccioneu qualitat 6 (la qualitat serà similar a la d'MP3 a 96 kbps).

FIGURA 1.41. Exportant àudio a OGG amb qualitat reduïda



Ja teniu la cançó preparada en els dos formats i amb el pes reduït en un ~70%.

1.2.3 Vídeo: codificació de vídeo, conversió de formats (exportar i importar)

A l'hora d'afegir fitxers de vídeo a una pàgina HTML us trobareu amb el mateix problema que amb l'àudio: cada navegador suporta uns formats o uns altres, i com els fitxers de vídeo també inclouen àudio comprimit en diferents formats és possible que un navegador pugui reproduir el vídeo però no l'àudio que inclou.

Encara que existeixen més formats, a la taula 1.17 només es mostren els que tenen una millor compatibilitat amb els navegadors més populars:

TAULA 1.17. Compatibilitat de formats de vídeo

Format	Còdec àudio	Còdec vídeo	Extensió	Chrome	Firefox	Internet Explor-er	Opera	Safari
WebM	Vorbis	VP8	webm	Sí	4.0	9.0*	10.60	3.1
MP4	AAC	H.264	MP4	Sí	Sí*	9.0	Sí	Sí
MP4	MP3	H.264	MP4	Sí	Sí*	9.0	Sí	3.1

El nombre correspon a la primera versió en què està disponible. * Indica que requereix algun connector o depèn del sistema operatiu.

Com es pot veure a la taula, tots els navegadors moderns suporten els formats WebM i MP4, encara que no sempre directament. Per exemple, Firefox depèn del sistema operatiu o el *hardware*, mentre que Internet Explorer 9 requereix un connector per reproduir el format WebM. Per altra banda, Google Chrome ho suporta a totes les seves versions, ja que es tracta d'un format desenvolupat per la mateixa companyia.

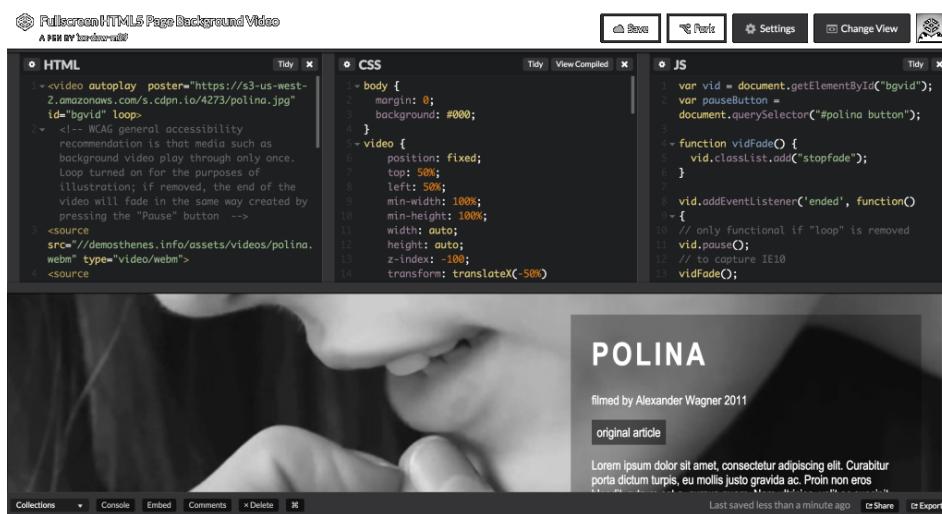
Un cop més, la solució és afegir el mateix fitxer en diferents formats dins de l'element video:

```

1 <video>
2   <source src="fitxer_video.mp4" type="video/mp4">
3   <source src="fitxer_video.webm" type="video/webm">
4   Ho sento, el teu navegador no suporta vídeo d'HTML5
5 </video>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/xZjRVN (recordeu que perquè funcioni heu de canviar el nom dels fitxers per dues URL a vídeos penjats a Internet).

Un exemple força més complex, que mostra com fer servir un vídeo com a fons d'una pàgina, el podeu trobar al següent pen (codepen.io/ioc-daw-m09/pen/PZdoqW) i el resultat es el que es pot apreciar a la figura 1.42.

FIGURA 1.42. Vídeo com a fons de la pàgina

Opcions de l'element "video"

De la mateixa manera que a l'element `audio`, es poden afegir atributs a l'element `video` per configurar-lo com es pot veure a la taula 1.18. La majoria d'atributs són compartits entre tots dos elements.

TAULA 1.18. Atributs de l'etiqueta "video"

Atribut	Efecte
<code>Autoplay</code>	El so es reproduirà tan aviat com sigui possible.
<code>Controls</code>	Es mostraran els controls de reproducció.
<code>Loop</code>	El vídeo es reproduirà des del principi en acabar.
<code>Muted</code>	El vídeo s'iniciarà silenciat, s'haurà de canviar l'estat via codi.
<code>SRC</code>	Si només es necessita un format de vídeo es pot fer servir aquest atribut en lloc de l'etiqueta <code>source</code> .
<code>Volume</code>	Volum amb el qual es reproduirà el vídeo, entre 0.0 (silenci) i 1.0 (molt alt)
<code>Height</code>	Alçada del reproductor de vídeo.
<code>Width</code>	Amplada del reproductor de vídeo.
<code>Poster</code>	URL d'una imatge que és mostrarà mentre el vídeo no comenci a reproduir-se.

Hi ha més atributs, però no tenen cap efecte.

Es pot trobar més informació sobre els atributs de vídeo en el següent enllaç: goo.gl/Tda7vd.

Generalment, no voldreu fer servir l'atribut `autoplay` perquè és força molest per als usuaris, tant si es tracta de vídeo com d'àudio.

En el cas dels fitxers de vídeo no s'acostuma a fer servir l'atribut `loop`, encara que hi ha excepcions; per exemple, si el vídeo es fa servir com a fons segurament voldrem repetir-lo indefinidament, però aquest ús està molt més estès en el cas de l'àudio.

Com que esteu forçats a emprar com a mínim dos formats de vídeo diferent per assegurar-vos que els vídeos es poden reproduir en una major varietat de navegadors, la propietat `src` tampoc s'usa habitualment.

1.2.4 Integració d'àudio i vídeo en una animació

No és gaire freqüent a l'hora de treballar amb vídeo afegir-lo sense fer cap modificació. Generalment, l'haureu d'editar per retallar fragments a l'inici i al final com a mínim, canviar de format, afegir-li una *intro* i/o *outro* i modificar l'àudio.

Encara que aquest tema no es tractarà en aquests materials, cal saber que un recurs força utilitzat a l'hora d'incloure música en un vídeo és fer servir fitxers d'àudio ja preparats per fer bucles (*loop*, en anglès); això permet fer servir el mateix àudio des del principi fins al final sense haver d'encadenar diverses cançons.

A continuació trobareu un exemple de com editar un vídeo incloent-hi una *intro* i substituint l'àudio fent servir **Avidemux**. En trobareu un tutorial al següent video:



<https://www.youtube.com/embed/1vltrj8Wsfl?controls=1>

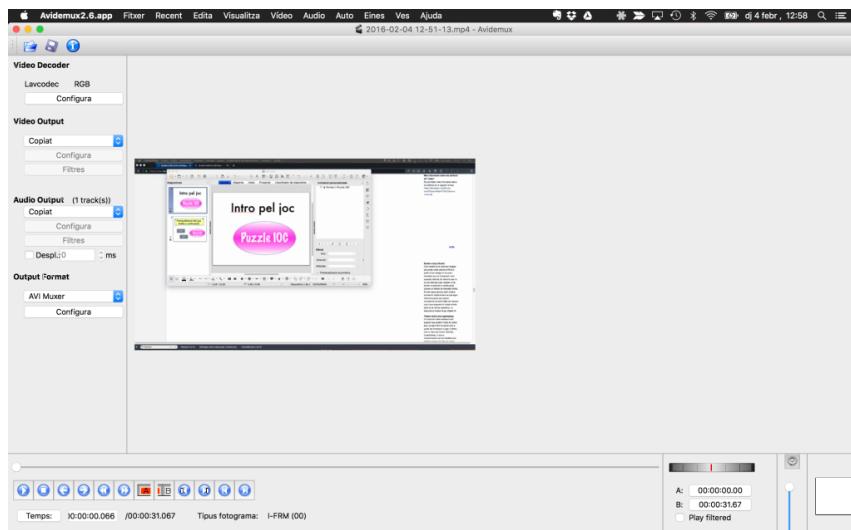
Creació d'un vídeo per al web amb Avidemux

Per a aquest exemple heu de fer servir dos vídeos enregistrats prèviament, ja sigui amb un programari com OBS o amb algun dispositiu (com pot ser un mòbil o una *webcam*).

El primer que heu de fer és obrir l'Avidemux. Si no el teniu instal·lat el podeu descarregar del següent enllaç: sourceforge.net/projects/avidemux.

Una vegada obert, veureu la pantalla que es mostra a la figura 1.43, on heu de seleccionar *Fitxer / Open* i triar el fitxer amb el vídeo que voleu fer servir com a introducció.

FIGURA 1.43. Fitxer obert a Avidemux



El primer pas consistirà a eliminar un fragment de vídeo que es trobi entre l'inici del vídeo i l'inici de la seqüència que voleu fer servir com a introducció. Per a això fareu servir els

botons de la barra de reproducció inferior fins a ajustar el *frame* exacte en el qual voleu iniciar l'acció. Feu servir el botó de reproducció (triangle) fins a arribar al fragment que us interessa, i llavors afineu la posició amb les fletxes dreta i esquerra per ajustar el *frame*, com es pot veure a la figura 1.44.

FIGURA 1.44. Barra de control de frames



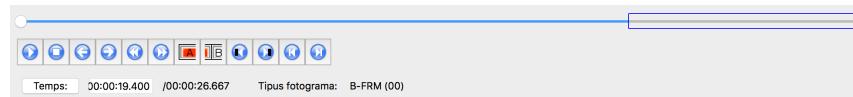
A continuació, cliqueu sobre el **botó de marcador B**, de manera que quedarà marcada tota la zona des del principi fins al *frame* seleccionat.

FIGURA 1.45. Rang de frames A seleccionat



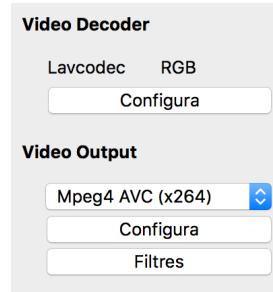
Per esborrar aquest fragment aneu al menú *Edit / Delete*, això farà que el vídeo comenci a partir d'aquest *frame*. Seguidament, repetiu els mateixos passos, però cercant l'últim *frame* del vídeo que voleu incloure com a introducció, i cliqueu el **botó de marcador A**, de manera que se seleccionarà tot el fragment des d'aquest *frame* fins al final del vídeo, com es pot apreciar a la figura 1.46.

FIGURA 1.46. Rang de frames B seleccionat



Esborreu-lo amb *Edit / Delete*, i ja gairebé tindreu el primer vídeo llest per afegir-lo com a introducció. Abans de desar-lo seleccioneu un còdec de sortida, per exemple Mpeg4 AVC (x264) com el que s'ha seleccionat a la figura 1.47, ja que, si no ho feu, us pot llençar un error perquè els *frames* clau poden no concordar amb la llargària del vídeo.

FIGURA 1.47. Selecció de codecs de sortida



Deseu-lo seleccionant *File / Save*. No cal canviar cap opció, feu servir el format per defecte i el nom Intro.avi.

Repetiu el procés amb el segon vídeo:

- Obriu-lo amb *File / Open*.

- Editeu el principi i el final del vídeo, seleccionant els fragments que vulgueu descartar i fent servir l'opció del menú *Edit / Delete*.
- Seleccioneu un còdec de sortida, per exemple Mpeg4 AVC (x264).
- Guardeu-lo amb *File / Save* amb el nom Principal.avi.

Una vegada editat el segon vídeo haureu de tenir els dos fitxers amb extensió .avi, que és el format per defecte amb el qual els desa Avidemux.

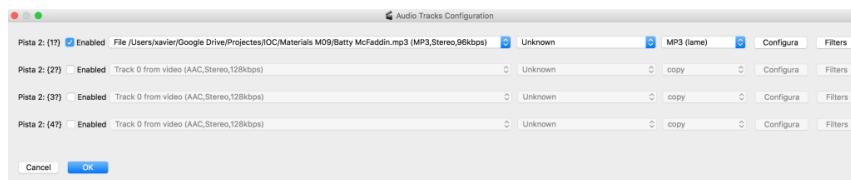
Per unir-los tots dos heu de seguir aquests passos:

1. Obriu a Avidemux el fitxer Intro.avi amb *File / Open*.
2. Seleccioneu l'opció del menú *File / Append* i afegiu Principal.avi.
3. Guardeu-lo amb *File / Save* i el nom Complet.avi.

Amb el vídeo complet editat i acoblat, només resta afegir el so. Com que no heu seleccionat cap còdec d'àudio de sortida en desar els vídeos editats anteriorment, el vostre vídeo no conserva les pistes d'àudio. A continuació afegireu una cançó (pròpia o amb llicència lliure) que prèviament hagueu preparat amb algun programari d'àudio, com per exemple Audacity.

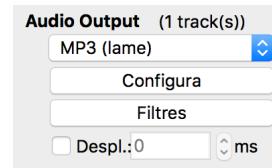
Al menú principal seleccioneu *Audio / Select Track*; per defecte estarà seleccionada la pròpia del vídeo (que en aquest cas no conté res). Si feu clic sobre el primer desplegable podreu navegar i afegir el vostre propi so, i fent clic sobre el tercer desplegable podreu seleccionar quin còdec d'àudio voleu fer servir; en aquest cas MP3, com es pot veure a la figura 1.48.

FIGURA 1.48



A sota de l'apartat *Audio Output* apareixerà *Codec MP3*, com es pot apreciar a la figura 1.49.

FIGURA 1.49. Configuració de pistes d'àudio



Finalment, deseu-lo seleccionant *File / Save* amb el nom Complet-amb-so.avi, i ja es pot donar per finalitzat el vostre vídeo.

1.2.5 Plataformes de vídeo en 'streaming'

Les dues plataformes més utilitzades per a la distribució de vídeo en *streaming* són YouTube i Vimeo. La primera és completament gratuïta, mentre que la segona

ofereix diferents plans als usuaris, oferint un compte gratuït però amb opció de pagar per accedir a millors.

Encara que inicialment només Vimeo incloïa l'opció d'ofrir continguts de pagament, YouTube ha afegit també aquesta opció, de manera que a totes dues plataformes és possible crear canals de pagament o pujar vídeos que només els usuaris que hagin pagat poden veure.

Per obtenir més informació sobre els continguts de pagament de YouTube es pot visitar aquest enllaç: goo.gl/LjSx9Y.

YouTube

YouTube és la plataforma més coneguda per a reproducció de vídeo en *streaming*, i en formar part del paquet d'aplicacions de Google Inc., qualsevol persona amb un compte de correu de Gmail és automàticament usuària de YouTube, amb permisos per crear el seu propi canal i pujar els seus propis continguts.

Un altre avantatge d'aquesta plataforma és que està molt integrada amb el mateix cercador de Google, i això fa que els vídeos que s'hi carreguen siguin molt més fàcils de trobar, perquè reben una major rellevància.



YouTube és la plataforma líder en continguts de vídeo. Font: YouTube

Cerca de tràiler d'una pel·lícula a Google

Proveu a cercar a Google "tràiler Star Wars".

Segons la versió del navegador, els resultats poden ser diferents, però com veieu a la figura 1.50 ens mostra primer un resultat d'un tràiler a YouTube com a suggeriment, sent els dos resultats següents també enllaços a YouTube, per sobre dels resultats de la distribuïdora de la pel·lícula.

FIGURA 1.50. Resultat de la cerca a Google de "tràiler 'Star Wars'"

Aproximadament 91.200.000 resultats (0,71 segons)

Star Wars: The Force Awakens Trailer (Official) - YouTube
<https://www.youtube.com/watch?v=sGbxmsDFVnE>

Suggeriments

Star Wars: The Force Awakens Official Teaser Trailer #1 ...
<https://www.youtube.com/watch?v=OMOVFvcNvE>
28 nov. 2014 - Penjat per Movieclips Trailers

Star Wars: Episode VII - The Force Awakens Official Teaser Trailer #1 ... In addition to being the #1 Movie ...
[Trailer #1](#)

Star Wars: El Despertar de la Fuerza – Tráiler Oficial ...
<https://www.youtube.com/watch?v=5eLdSDGh9k>
19 oct. 2015 - Penjat per Star Wars Latinoamérica

Google mostra els resultats de YouTube fins i tot per sobre dels corresponents a la distribuïdora de la pel·lícula.

Un altre avantatge és que l'equip d'enginyers de Google proporciona l'SDK (Software Development Kit) per a múltiples plataformes, de manera que és relativament senzill integrar la reproducció de vídeos de YouTube en les nostres pròpies aplicacions per a qualsevol dispositiu.

Per altra banda, en els últims temps YouTube ha afegit opcions per pujar vídeos amb visió de 360º (com el que es pot veure a la figura 1.51) que permeten als usuaris veure al voltant del punt on es troba la càmera:



<https://www.youtube.com/embed/aQd41nbQM-U?controls=1>

També existeix l'opció de pujar continguts enregistrats per ser visualitzats amb ulleres de realitat virtual:



<https://www.youtube.com/embed/u0hmMt5znX0?controls=1>

FIGURA 1.51. Vídeo 360º: Dreams of Dalí



Podeu trobar aquest vídeo en el següent enllaç:
<https://www.youtube.com/watch?v=F1eLelocAcU>.

Formats i resolucions recomanats per a YouTube Quan enregistrem un vídeo amb intenció de pujar-lo a YouTube s'ha de tenir en compte quines són les resolucions i els formats recomanats:

- **format vídeo:** MP4
- **còdec d'àudio:** AAC-LC
- **còdec de vídeo:** H.264

En cas de carregar un vídeo a YouTube amb una ràtio diferent de 16:9, s'afegeixen dues barres negres als costats.

Per determinar el tipus de vídeo és molt habitual referir-se només als píxels d'alçada, ja que la resolució total es pot extrapolar sabent que la ràtio amb la qual s'acostuma a treballar és 16:9 (correspondent a amplada:alçada). Com a orientació es pot fer servir la taula 1.19.

TAULA 1.19. Recomanació de 'bitrate' atesa la resolució i els quadres per segon a reproduir

Tipus	Resolució	Freqüència de bits estàndard (24, 25, 30)	Freqüència de bits alta (48, 50, 60)
2.160 p (4k)	3.840x2.160	35-45 Mbps	53-68 Mbps
1.440 p (2k)	2.560x1.440	16 Mbps	24 Mbps
1.080 p (Full HD)	1.920x1.080	8 Mbps	12 Mbps
720 p (HD)	1.280x720	5 Mbps	7,5 Mbps
480 p	845x480	2,5 Mbps	4 Mbps
360 p	640x360	1 Mbps	1,5 Mbps

I a la taula 1.20 es mostra la recomanació per a àudio:

TAULA 1.20. Recomanació de 'bitrate' per a àudio

Tipus	Freqüència de bits d'àudio
Mono	128 kbps
Estèreo	384 kbps
5.1	512 kbps

Es pot trobar tota la informació sobre la configuració recomanada en el següent enllaç: goo.gl/0Vlk4zY.

A continuació veureu un exemple de com pujar els vostres vídeos a YouTube i com incrustar-los en una pàgina.

Com pujar un vídeo a YouTube

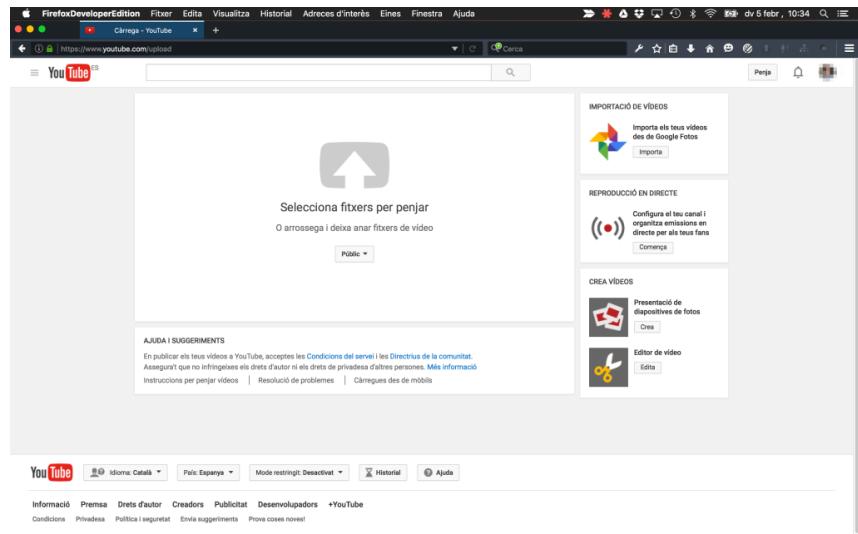
Aquesta és una tasca força habitual, ja que es poden pujar vídeos per compartir amb altres companys, per promocionar-vos, per als vostres clients o per a la vostra empresa, ja que YouTube és generalment la primera opció en el moment de publicar continguts de vídeo.

Primer de tot, heu de connectar amb el vostre compte de Google, si no hi esteu ja connectats, des del següent enllaç: accounts.google.com/ServiceLogin?service=youtube.

Una vegada connectats cliqueu al botó *Penja* que trobareu a la part superior dreta de la pantalla (vegeu la figura 1.52).

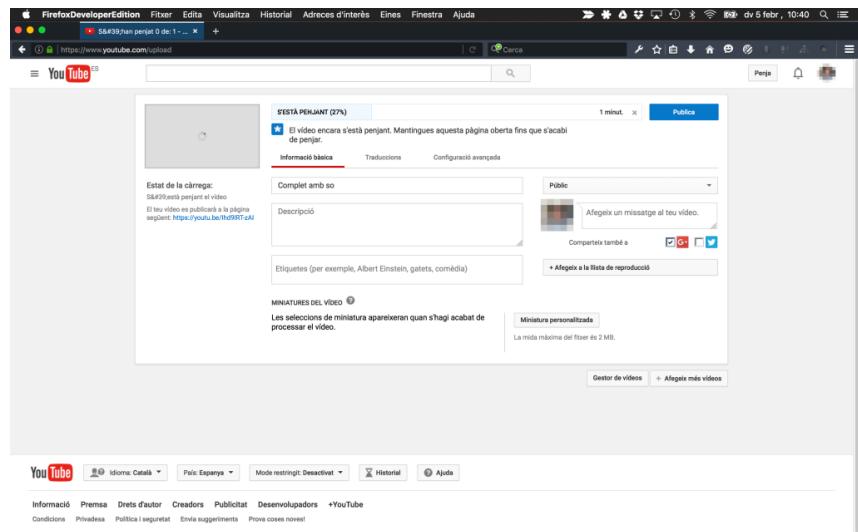
FIGURA 1.52. Barra d'accions de YouTube

A continuació, arrossegueu sobre la part central el vídeo que vulgueu penjar (vegeu la figura 1.53).

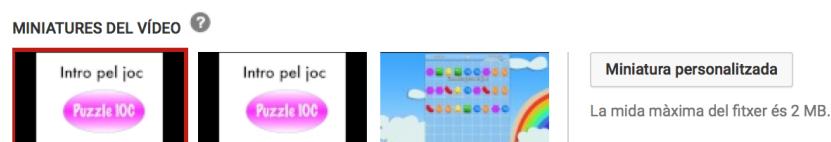
FIGURA 1.53. Carrega de fitxers a YouTube

Mentre es carrega, podreu afegir informació addicional sobre el vídeo com es pot apreciar a la figura 1.54, per exemple:

- el títol
- la descripció
- etiquetes per facilitar les cerques
- la privadesa

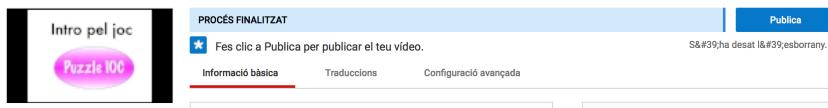
FIGURA 1.54. Entrada de dades adicionals d'un fitxer pujat a YouTube

Una vegada finalitzi la càrrega, podeu escollir quina és la imatge que voleu que es mostri com a miniatura (vegeu la figura 1.55).

FIGURA 1.55. Selecció de la miniatura a mostrar pel vídeo

Per acabar només cal clicar sobre el botó *Publica* per publicar el vídeo (vegeu la figura 1.56).

FIGURA 1.56. Procés de càrrega de vídeo finalitzat



Amb aquest últim pas, el vídeo ja es trobarà a YouTube (vegeu la figura 1.57).

FIGURA 1.57. Vídeo carregat i enllaç disponible



S'ha de tenir en compte que segons la mida del fitxer aquest pot no estar disponible immediatament, o està limitat pel que fa a les opcions de resolució. Una vegada els servidors de YouTube acaben de processar-lo, el vídeo es podrà reproduir correctament i rebreu un correu confirmant-vos que el processament ha finalitzat.

Vimeo

Atès que actualment tant YouTube com Vimeo ofereixen l'opció de cobrar als usuaris per visualitzar els vídeos, la diferència més gran entre aquestes plataformes és que amb Vimeo tant la qualitat dels vídeos com la quantitat de dades que s'hi poden pujar estan limitades i només els usuaris del compte més car poden cobrar per la visualització dels seus continguts.



Així doncs, encara que la base d'usuaris de Vimeo és molt gran, sembla una millor opció fer servir YouTube per als vostres vídeos o fer servir totes dues, ja que YouTube té menys limitacions. Tot i així, Vimeo ofereix una opció que no inclou YouTube (en el moment de la redacció d'aquests materials): és possible limitar la reproducció dels vídeos incrustats a un domini, de manera que no poden enllaçar als vostres vídeos per reproduir-los en pàgines alienes.

1.2.6 Animacions d'imatges i text

Fins fa uns anys calia utilitzar connectors de tercers per poder reproduir vídeos, afegir algun efecte als títols d'una pàgina web o crear jocs per al web. En alguns casos, fins i tot s'utilitzaven GIF animats per afegir algun tipus d'efecte, amb les limitacions que això comporta (pocs colors i no poder controlar la reproducció).

Afortunadament, avui dia, gràcies a les funcionalitats afegides a HTML i CSS, és possible crear tot tipus d'animacions d'imatges i texts fent servir les característiques natives dels navegadors i el llenguatge de programació JavaScript.

Un gran avantatge de fer servir JavaScript i CSS és que com que no es requereix cap connector per funcionar en els navegadors qualsevol aplicació o lloc web desenvolupat amb aquest llenguatge pot ser visualitzat per tots els usuaris, sense importar si estan connectats des d'un mòbil, un ordinador o un televisor intel·ligent.

Ús de fulls d'estil a les animacions

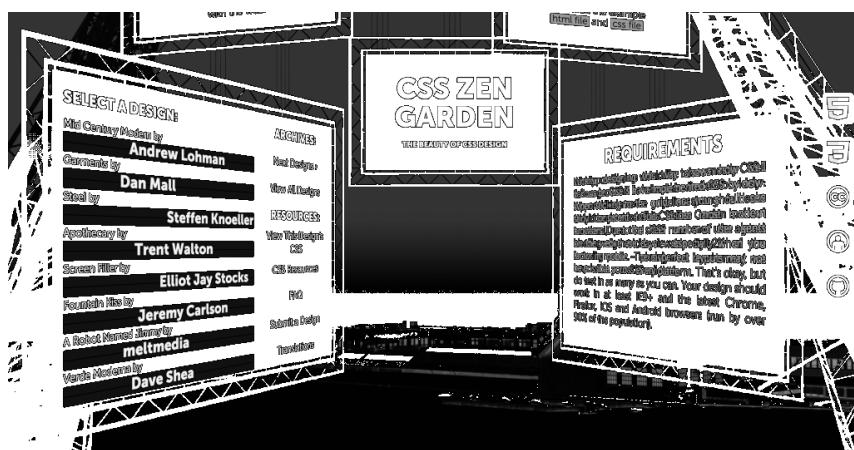
L'ús de CSS i CSS3 no tan sols ha permès separar l'estil del contingut de les pàgines HTML, sinó que també les ha enriquit, la qual cosa ha possibilitat la inclusió d'animacions que abans requerien l'ús de connectors com Flash Player o de JavaScript.

CSS Zen Garden

CSS Zen Garden (www.csszengarden.com) és un lloc web dedicat a la demostració de les possibilitats que ofereix CSS juntament amb HTML5. Per participar només cal descarregar el fitxer HTML i CSS i modificar el CSS al nostre gust. Al lloc web es poden veure centenars d'exemples de la mateixa pàgina HTML només modificant el fitxer CSS.

A diferència d'altres llocs, a Zen Garden no es fan servir les últimes novetats en CSS, sinó les que són compatibles amb la gran majoria de navegadors i es poden utilitzar de manera força segura (vegeu la figura 1.58).

FIGURA 1.58. Exemple d'estil aplicat amb CSS



El full aplicat inclou a més animacions, que es poden provar a <http://www.csszengarden.com/219/>.

Per una banda, es troben les **transicions** que permeten suavitzar els canvis produïts en un element en canviar les seves propietats, per exemple la seva posició, l'opacitat o la mida. Si no s'apliquen transicions, aquests canvis són instantanis, però aplicant-les es creen automàticament totes les posicions intermèdies.

I per una altra, es disposa de les **animacions** creades amb CSS3 amb una sèrie de **frames** clau per definir diferents estats d'un element al llarg de la seva duració.

Fixeu-vos en el següent exemple, en passar el cursor per sobre del títol canvia el color de la font i del fons:

```

1 <style>
2 .title {
3   text-align: center;
4   margin: 20px;

```

```
5   font-weight: bold;
6   font-size: 3em;
7 }
8
9 .title:hover {
10  background-color: #2897E8;
11  color:#ffff;
12 }
13 </style>
14 <div class="title">Institut Obert de Catalunya</div>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/xZaGbW.

Com que el canvi és instantani, la sensació que dóna és de brusquedat.

Ara proveu el següent, en el qual s'ha afegit a la classe CSS una transició de 2 segons:

```
1 <style>
2 .title {
3   transition: all 2s ease;
4   text-align: center;
5   margin: 20px;
6   font-weight: bold;
7   font-size: 3em;
8 }
9
10 .title:hover {
11  background-color: #2897E8;
12  color:#ffff;
13 }
14 </style>
15 <div class="title">Institut Obert de Catalunya</div>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/XXPbdz.

Només s'ha afegit una línia, però la sensació ara és molt diferent: fins que no han passat els 2 s no es completa el canvi.

Aprofitant aquestes transicions, es poden afegir més efectes de CSS, per exemple canviar l'opacitat:

```
1 <style>
2 .title {
3   transition: all 2s ease;
4   text-align: center;
5   margin: 20px;
6   font-weight: bold;
7   font-size: 3em;
8 }
9
10 .title:hover {
11  opacity:0.1;
12 }
13 </style>
14 <div class="title">Institut Obert de Catalunya</div>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/MKqwJo.

O canviar la mida:

```

1 <style>
2   .title {
3     transition: all 2s ease;
4     text-align: center;
5     margin: 20px;
6     font-weight: bold;
7     font-size: 3em;
8   }
9
10  .title:hover {
11    font-size: 6em;
12  }
13 </style>
14 <div class="title">Institut Obert de Catalunya</div>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/wMEaJK.

Fins ara s'ha fet servir la funció d'interpolació `ease`, però n'hi ha més que es poden fer servir, com per exemple `ease-in` i `ease-out` (vegeu la taula 1.21).

TAULA 1.21. Funcions de transició

Funció	Efecte
<code>Ease</code>	Comença i acaba a poc a poc.
<code>Ease-in</code>	Comença a poc a poc.
<code>Ease-out</code>	Acaba a poc a poc.
<code>Lineal</code>	Sense acceleració.

Encara que hi ha més funcions, no totes són compatibles amb tots els navegadors.

Existeixen altres funcions de transició, i fins i tot se'n poden definir de noves, però aquesta especificació encara es troba en fase experimental. Podeu trobar més informació sobre les transicions a: goo.gl/1c4k0h.

Vegeu la diferència en aquest exemple; cada títol fa servir una funció de transició diferent:

```

1 <style>
2 div {
3   text-align: center;
4   margin: 20px;
5   font-weight: bold;
6   font-size: 2em;
7   width: 30%;
8   float: left;
9 }
10
11 div:hover {
12   background-color: #2897E8;
13 }
14
15 .title1 {
16   transition: all 2s ease;
17   color: green;
18 }
19
20 .title2 {
21   transition: all 2s ease-in;
22   color: blue;
```

```

23 }
24
25 .title3 {
26   transition: all 2s ease-out;
27   color: red;
28 }
29 </style>
30 <div class="title1">Institut Obert de Catalunya</div>
31 <div class="title2">Institut Obert de Catalunya</div>
32 <div class="title3">Institut Obert de Catalunya</div>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/dGqoJL.

Com ja heu vist, es poden animar propietats; per tant, una possibilitat és animar propietats de posició, com per exemple `left`, per a un element amb `position:absolute`:

```

1 <style>
2 div {
3   position: absolute;
4   left: 0;
5   transition: all 0.5s ease-out;
6   margin: 20px;
7   font-weight: bold;
8   font-size: 2em;
9 }
10
11 div:hover {
12   left: 50px;
13 }
14 </style>
15 <div class="title">Institut Obert de Catalunya</div>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/LGJROz.

Encara que això funciona, obliga a establir per a cada element l'atribut `position`, que altera la manera com es col·loca l'element dins de la pàgina i que pot tenir efectes no desitjats. Una solució molt més apropiada és fer servir `transform:translate(x,y)`, que permet definir un desplaçament horitzontal i vertical sense modificar aquest atribut.

```

1 <style>
2 div {
3   transition: all 0.5s ease-out;
4   margin: 20px;
5   font-size: 1.5em;
6 }
7
8 .title {
9   font-weight: bold;
10  font-size: 2em;
11 }
12
13 div:hover {
14   transform: translate(50px, 10px);
15 }
16 </style>
17
18 <div class="title">Institut Obert de Catalunya</div>
19 <div>L'institut que va on tu vas</div>
```

Propietat 'transform'

Podeu trobar una llista completa de les possibilitats d'aquesta propietat en el següent enllaç: goo.gl/gZZhwT.

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/wMEJXZ.

Com podeu veure en aquest exemple, l'efecte de translació s'aplica sobre la posició original de cada element, i no cal establir la seva posició absoluta, la qual cosa el fa molt més versàtil. En cas de necessitar només desplaçar en horitzontal o vertical, també disposem de les propietats `transform:translateX(x)` o `transform:translateY(y)`.

La propietat `transform` de CSS3 ens permet fer diverses **transformacions**, com traslladar, rotar o escalar, però és una tecnologia experimental. S'ha de comprovar la compatibilitat amb navegadors, i en alguns casos s'ha de fer servir un prefix (`-moz-`, `-webkit-`, `-ms-` o `-o-`) per poder fer-les servir.

Una altra opció que ofereix la propietat `transform` és escalar un element, sigui per augmentar-ne la mida o per reduir-la, com es pot veure en el següent exemple:

```

1 <style>
2   img {
3     transition: all 2s ease;
4     display: block;
5     margin: 50px auto;
6   }
7
8   img:hover {
9     transform:scale(2);
10  }
11
12 </style>
13
14 
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/LGJWMQ.

De la mateixa manera que succeeix amb `transform:translate()`, existeixen dues funcions diferents per escalar només l'eix X (`transform:scaleX(x)`) o només l'eix Y (`transform:scaleY(y)`); encara que es vol escalar l'un i l'altre en diferents proporcions, també es pot fer servir `transform:scale(x, y)`.

També és possible fer rotacions, tant en dues com en tres dimensions. En aquest primer exemple podeu veure una rotació en dues dimensions:

```

1 <style>
2   img {
3     transition: all 2s ease;
4     display: block;
5     margin: 50px auto;
6   }
7
8   img:hover {
9     transform:rotate(30deg);
10  }
11
12 </style>
13
14 
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/zrJZXz.

Les rotacions s'indiquen en graus (*degrees* en anglès).

A diferència de les altres transformacions, `transform:rotateX(x)` i `transform:rotateY(y)` no s'apliquen a l'eix en dues dimensions, sinó en tres. Vegeu aquest exemple:

```

1 <style>
2   img {
3     transition: all 1s ease;
4     display: block;
5     margin: 50px auto;
6   }
7
8   img:hover {
9     transform:rotateX(180deg);
10  }
11
12 </style>
13
14 
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/qbMrzx.

Aquestes transformacions (vegeu la taula 1.22) poden combinar-se per crear animacions més complexes com la que podeu veure a continuació:

```

1 <style>
2   img {
3     transition: all 1s ease;
4     display: block;
5     margin: 50px auto;
6   }
7
8   img:hover {
9     transform: rotate(10deg) scale(1.5) translate(20px, 20px);
10  }
11
12 </style>
13
14 
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/WrgjGr.

TAULA 1.22. Resum de funcions de transformació

Funció	Efecte	Exemple
<code>translate()</code>	Trasllada l'element des del seu punt d'origen fins a $(+x,+y)$ unitats	<code>transform:translate(50px, 10px)</code>
<code>translateX()</code>	Trasllada l'element horitzontalment	<code>transform:translateX(50px)</code>
<code>translateY()</code>	Trasllada l'element verticalment	<code>transform:translateY(50px)</code>
<code>scale()</code>	Escala l'element horitzontalment i verticalment; si no s'especifica el segon paràmetre, s'aplica el mateix valor a les dues dimensions	<code>transform:scale(2,1.5).</code>
<code>scaleX()</code>	Escala l'element horitzontalment	<code>transform:scaleX(2)</code>

TAULA 1.22 (continuació)

Funció	Efecte	Exemple
scaleY()	Escala l'element verticalment	transform:scaleY(2)
rotate()	Rota l'element en dues dimensions	transform:rotate(30deg)
rotateX()	Rota l'element en l'eix X en tres dimensions	transform:rotateX(30deg)
rotateY()	Rota l'element en l'eix Y en tres dimensions	transform:rotateY(30deg)
rotateZ()	Rota l'element en l'eix Z en tres dimensions	transform:rotateZ(30deg)

Aquesta llista no és exhaustiva, només es mostren algunes de les transformacions vistes.

Es pot trobar la llista completa de funcions de filtre i la compatibilitat amb navegadors en el següent enllaç: goo.gl/bZtPhi.

Una altra propietat molt interessant de CSS3, tant per animar com per aplicar sense animacions, és la propietat `filter` que permet aplicar diferents filtres especials a l'element com poden ser esborronar, augmentar el contrast, afegirombres, canviar la imatge a blanc i negre o aplicar un color sèpia, entre d'altres (vegeu taula 1.23).

La propietat `filter` es troba en estat experimental i s'ha de comprovar la seva compatibilitat amb navegadors i la necessitat de prefixos abans d'utilitzar-la. En el cas del navegador Chrome s'ha d'afegir el prefix `-webkit-` per fer que funcioni correctament. Mozilla Firefox, en canvi, la reconeix tant amb el prefix `-webkit-` com sense cap prefix.

```

1 <style>
2 img {
3   transition: all 1s ease;
4   display: block;
5   margin: 20px;
6   float: left;
7 }
8
9 .un:hover {
10   -webkit-filter: blur(5px);
11 }
12
13 .dos:hover {
14   -webkit-filter: grayscale(100%);
15 }
16
17 .tres:hover {
18   -webkit-filter: sepia(100%);
19 }
20
21 .quatre:hover {
22   -webkit-filter: contrast(4);
23 }
24
25 .cinc:hover {
26   -webkit-filter: hue-rotate(120deg);
27 }
28
29 .sis:hover {
30   -webkit-filter: invert(1);
31 }
32 </style>
33 
34 
35
36

```

```

37 
38
39 
40
41 
42
43 
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/GoXmrB.

TAULA 1.23. Resum de funcions de filtres

Funció	Efecte	Exemple
grayscale()	Converteix l'element a escala de grisos	filter:grayscale(100%)
blur()	Esborrona l'element	filter:blur(5px)
sepia()	Tinta l'element de color sèpia	filter:sepia(100%)
contrast()	Augmenta el contrast de l'element	filter:contrast(4)
invert()	Inverteix els colors de l'element	filter:invert(1)

Aquesta llista no és exhaustiva, només es mostren alguns dels filtres a tall d'exemple.

Una altra opció molt potent que ofereix CSS3 és la possibilitat de crear animacions definint l'estat d'un element en diversos punts de l'animació.

```

1 <style>
2 div {
3   position: absolute;
4   text-align: center;
5   padding: 50px;
6   width: 40px;
7   font-size: 2em;
8   color: #fff;
9   font-weight: bold;
10  background: #2897E8;
11
12  /* animació */
13  animation-duration: 5s;
14  animation-name: rebot;
15  animation-iteration-count: infinite;
16 }
17
18 @keyframes rebot {
19   from {
20     left: 0;
21   }
22   50% {
23     left: 500px;
24     transform: rotateY(180deg) scale(2);
25     -webkit-filter: blur(5px);
26   }
27   to {
28     left: 0;
29   }
30 }
31 </style>
32 <div>IOC</div>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/BjORmo.

Per crear una **animació** cal definir com a mínim la seva duració amb l'atribut `animation-duration` i el seu nom amb `animation-name`. Si voleu que l'animació es repeteixi més d'una vegada heu d'especificar el nombre d'iteracions amb l'atribut `animation-iteration-count` com a nombre, o com a `infinite` si voleu un bucle infinit.

A continuació s'han de definir els *frames* clau de l'animació fent servir `@keyframes nom_animacio`, i a continuació, entre claus, els estats inicial (`from`) i final (`to`), i tots els estats intermedis que es necessitin. Aquests estats intermedis es defineixen indicant el percentatge d'animació amb el qual voleu definir aquest estat; per exemple: si indiqueu 50%, aquest serà l'estat a la meitat de l'animació.

Preprocessadors: LESS i SASS

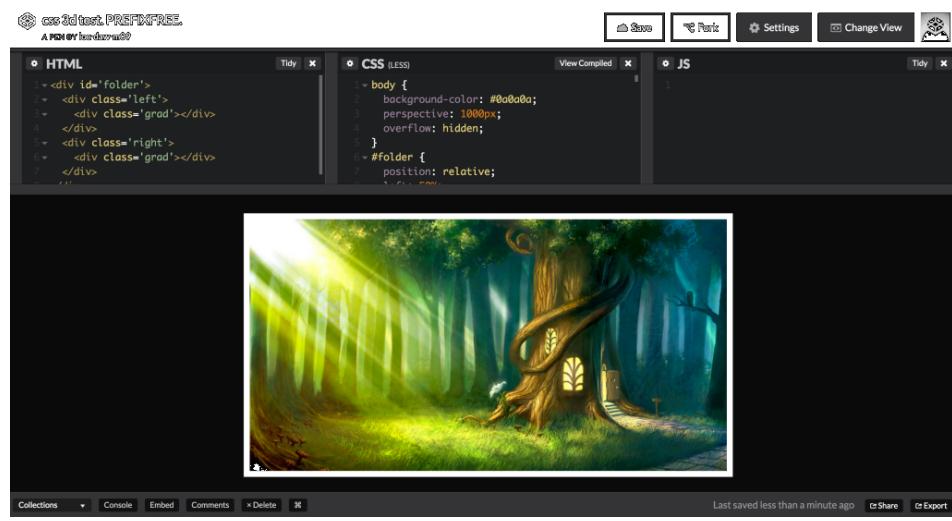
A causa de les limitacions de CSS, és molt freqüent l'ús de preprocessadors com LESS i SASS, que permeten fer servir variables, jerarquies de classes i funcions combinades amb CSS clàssic per simplificar el codi.

Després, aquests fitxers són processats i es genera el codi CSS que s'inclou a la pàgina.

Per acabar amb aquesta secció podeu veure aquests exemples més elaborats de l'ús de CSS3, creats per diversos autors a CodePen:

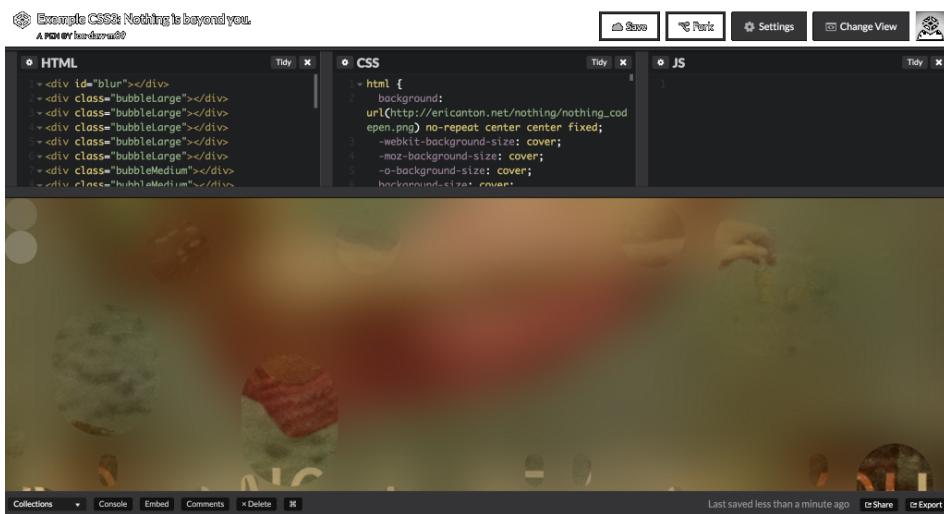
- **Transformació 3D:** la figura 1.59 mostra una fotografia mig plegada que s'obre en passar el cursor per sobre. En aquest cas s'ha fet servir el preprocessador **LESS**; per veure el codi CSS compilat heu de prémer el botó *View Compiled*.

FIGURA 1.59. CSS Transformació 3D



Podeu trobar el codi d'aquest pen a <http://codepen.io/ioc-daw-m09/pen/zrJxqq>.

- **Animació de posició, mida i opacitat:** aquest exemple mostra com fent servir només CSS es poden animar els elements HTML de la pàgina per crear diferents efectes, en aquest cas bombolles submarines (vegeu la figura 1.60).

FIGURA 1.60. Animació de posició, mida i opacitat


The screenshot shows a CodePen interface with three tabs: HTML, CSS, and JS. The CSS tab contains the following code:

```

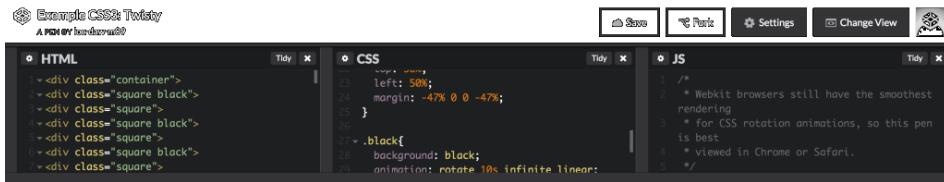
1 html {
2   background:
3     url('http://ericanton.net/nothing/nothing_codepen.png') no-repeat center center fixed;
4   -webkit-background-size: cover;
5   -moz-background-size: cover;
6   -o-background-size: cover;
7   background-size: cover;

```

The preview window shows a blurred background image of bubbles with varying opacities and sizes.

Podeu trobar el codi d'aquest pen a <http://codepen.io/ioc-daw-m09/pen/YwOPRj>.

- **Animació de rotació:** podeu veure a continuació com fent servir només quatre declaracions CSS és possible crear patrons d'animació molt interessants (vegeu la figura 1.61).

FIGURA 1.61. Animació de rotació


The screenshot shows a CodePen interface with three tabs: HTML, CSS, and JS. The CSS tab contains the following code:

```

1 .square {
2   left: 30%;
3   margin: -47% 0 -47% 0;
4 }
5
6 .block{
7   background: black;
8   animation: rotate 10s infinite linear;

```

The preview window shows a complex, spiraling pattern of overlapping squares rotating around a central point.

Podeu trobar el codi d'aquest pen a <http://codepen.io/ioc-daw-m09/pen/gPdbqE>.

Animacions amb JavaScript

Encara que JavaScript es pot fer servir per incloure algunes animacions a una pàgina web, habitualment no és el més indicat, ja que moltes vegades aplicar el mateix efecte amb CSS3 produeix un millor efecte.

jQuery

jQuery és una llibreria gratuïta molt utilitzada que permet portar a terme les accions més habituals en JavaScript d'una manera molt més simple. En podeu trobar més informació a: jquery.com.

Es pot trobar una introducció a jQuery en l'apartat "Creació i integració d'elements multimèdia al web" de la unitat "Creació i integració d'elements multimèdia al web".

JavaScript pla (Plain JavaScript)

No es tracta d'una versió diferent de JavaScript, sinó que, per evitar confusions, es pot fer servir el terme JavaScript pla per indicar que un codi en JavaScript no inclou referències a cap llibreria externa. Per exemple, en aquesta secció trobeu un exemple amb JavaScript pla i l'altre amb jQuery (que també és JavaScript però fent servir la llibreria jQuery).

Compareu els següents tres exemples que realitzen la mateixa animació fent servir diferents mètodes: **JavaScript pla**, **CSS3** i **jQuery**.

El primer amb JavaScript pla:

```

1 <style>
2   #quadre {
3     position: absolute;
4     width: 100px;
5     height: 100px;
6     background: red
7   }
8 </style>
9
10 <div id="quadre">
11 </div>
12
13 <script>
14   var quadre = document.getElementById("quadre"),
15     velocitat = 10, // nombre de píxels que es desplaça per moviment
16     posx = 0, // posició inicial
17     direccio = 1, // quan és positiu es mou cap a la dreta
18     freq = 50; // nombre de ms entre moviment i moviment
19
20   setInterval(function() {
21     if (posx >= 500) {
22       direccio = -1;
23     } else if (posx <= 0) {
24       direccio = 1;
25     }
26
27     posx += velocitat * direccio;
28     quadre.style.left = posx + "px";
29
30   }, freq);
31 </script>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/wMjGmy.

Aquí podeu veure el mateix efecte però fent servir animacions de CSS3:

```

1 <style>
2   div {
3     position: absolute;
4     width: 100px;
5     height: 100px;
6     background: red;
7
8     /* animació */
9     animation-duration: 2.5s;
10    animation-name: rebot;
11    animation-direction: alternate;
12    animation-iteration-count: infinite;
13  }
14
15  @keyframes rebot {
16    from {
17      left: 0px;
18    }
```

```
19   to {
20     left: 500px;
21   }
22 }
23 </style>
24
25 <div></div>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/gPzroR.

El codi és força més clar, i en executar-lo l'animació és molt fluida, ja que fa servir transicions (o interpolacions de moviment) per suavitzar el moviment entre els punts inicial i final.

I finalment, el mateix exemple fent servir jQuery:

```
1 <style>
2   #quadre {
3     position: absolute;
4     width: 100px;
5     height: 100px;
6     background: red;
7   }
8 </style>
9
10 <div id="quadre">
11 </div>
12
13 <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js" type="text/javascript">
14 </script>
15
16 <script>
17   var desti = 500;
18
19   function rebot() {
20     $('#quadre').animate({
21       left: desti + 'px'
22     }, 2500, function() {
23       if (desti > 0) {
24         desti = 0;
25       } else {
26         desti = 500;
27       }
28       rebot(); // Es torna a cridar a si mateixa
29     })
30   };
31
32   rebot(); // Iniciem la seqüència
33 </script>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/pgVyWO.

Aquesta versió de l'exemple és més concisa que la versió en JavaScript, ja que només hem de controlar una variable i la resta es fa internament. També inclou la interpolació de moviment com el CSS, de manera que el moviment és més suau que en el cas de JavaScript pla.

Per poder fer servir la biblioteca **jQuery** aquesta s'ha de carregar primerament, mentre que si s'utilitza només CSS3 i/o JavaScript no és necessari carregar cap recurs extern.

GIF animats

Una altra opció a l'hora de mostrar animacions en una pàgina web és fer servir un format d'imatge que suporti animacions. Actualment el seu ús està molt limitat, ja que **l'únic format d'imatges animades suportat nadiuament per tots els navegadors és el format GIF**, i aquest presenta unes limitacions importants:

- Està limitat a un **màxim de 256 colors**.
- **No suporta transparències parcials**, fet que ens obliga a fer servir un fons de color pla igual tant per a l'element en què s'incrustarà la imatge com per a la imatge.

GIF animats a les xarxes socials

Xarxes socials com Facebook i Twitter ofereixen suport per a GIF animats, fet que ha obert noves oportunitats per fer servir aquest format.

Això ha fet que, encara que continua sent un format molt popular per afegir animacions a les xarxes socials i als fòrums, el seu ús professional sigui força escàs.

L'ús més clàssic que es pot donar als GIF animats és la creació de bànders publicitaris, encara que darrerament ha estat desbancat per l'ús de JavaScript i CSS3 per fer aquestes animacions, ja que permeten personalitzar els anuncis i afegir-hi més característiques. Tot i així, aquesta és una forma molt simple de crear un bànder animat, i en els casos en què no necessiteu gaire complexitat pot ser una solució, perquè només l'heu d'incrustar com una imatge dins d'un enllaç al lloc que anuncia, i ja ho teniu:

```

1 <a href="http://ioc.xtec.cat/">
2   
3 </a>
```

Les imatges en CodePen es poden incrustar com a dades en base64, en lloc de fer servir un URL.

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/XXqdBv.

Cal tenir en compte que encara que es poden fer servir **GIF animats** des dels anys noranta, l'estil ha canviat molt. Els GIF antics en general no són acceptables per a usos professionals i s'han d'evitar. Per exemple, la típica bústia animada.

Un altre dels casos en què us pot interessar fer servir un GIF animat és per mostrar un indicador de progrés o donar retroacció quan l'usuari porta a terme alguna acció a la vostra pàgina. Per exemple, si en clicar un botó es processa alguna informació o es fa una petició al servidor, és interessant mostrar algun tipus d'indicació que s'està fent alguna cosa.

Com que els correus electrònics es poden enviar en format HTML i incloure imatges, un altre ús que li podeu donar als GIF animats és afegir animacions als correus. Tenint en compte que el suport per a CSS3 no està tan estès com el suport per als GIF animats, és una bona solució si es vol afegir alguna animació.

A l'hora d'afegir un GIF animat a un **correu comercial** s'ha de tenir molt en compte que no tots els navegadors reproduueixen les animacions, així que s'ha de pensar molt bé quin serà el primer *frame*, perquè és possible que sigui l'únic que es visualitzi.

Hi ha moltes alternatives a l'hora de crear un GIF animat. Les dues més habituals són: la creació a partir d'un vídeo ja existent i la creació a partir d'una seqüència d'imatges.

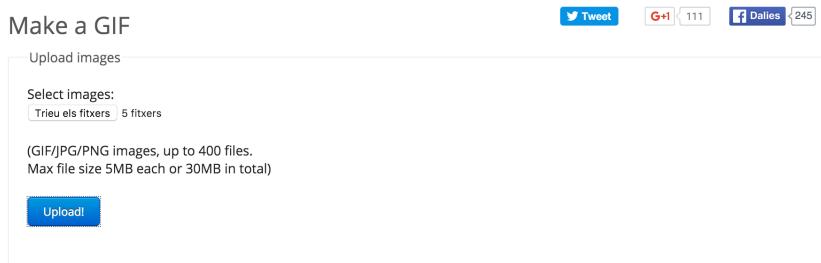
Creació d'un GIF animat a partir d'una seqüència d'imatges en línia

En aquest exemple veureu com crear un GIF animat fent servir les eines de la pàgina ezgif.com.

Una vegada tingueu les imatges que voleu fer servir en el vostre GIF heu d'anar a ezgif.com/maker i seguir els passos següents:

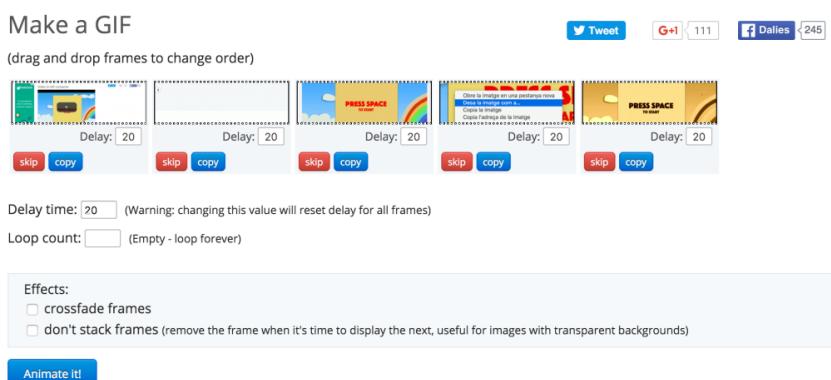
1. Trieu les imatges que voleu utilitzar i cliqueu en el botó *Upload!* (vegeu la figura 1.62).

FIGURA 1.62. Pantalla principal de ezgif.com



2. Quan estiguin carregades podreu veure les opcions per a cada imatge: el *delay* (temps fins al següent *frame*), si la voleu descartar (*skip*) o duplicar (*copy*). L'opció del temps d'espera es pot especificar globalment en la secció inferior, on també podeu indicar el nombre de vegades que vulgueu que es repeteixi l'animació; indefinidament, si ho deixeu en blanc (vegeu la figura 1.63).

FIGURA 1.63. Imatges pujades a ezgif.com per crear un GIF



CSS3 als clients de correu electrònic

Es pot comprovar la compatibilitat de diversos clients de correu electrònic en el següent enllaç:
www.campaignmonitor.com/css/

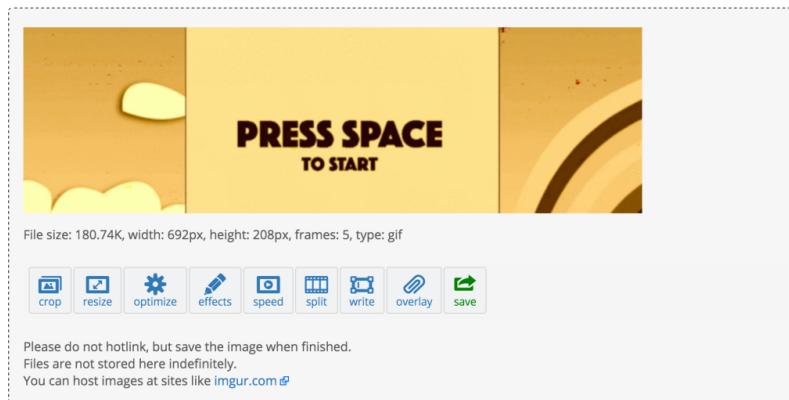
'Frames'

En el context dels GIF animats, i generalment sempre que parlem d'animacions, el terme *frame* es refereix a cadascuna de les imatges que formen l'animació.

3. En el moment que estigueu conformes amb la composició heu de fer clic al botó *Animate it!* per generar el GIF animat.
4. Amb el vostre GIF generat podreu fer diverses modificacions (vegeu la figura 1.64) fent servir la barra d'eines: afegir efectes, canviar la mida, modificar la velocitat de reproducció, rotar la imatge, etc.

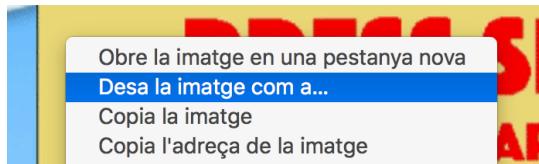
FIGURA 1.64. Opcions d'edició del GIF generat

Animated gif:



5. Per guardar el GIF modificat només heu de fer clic amb el botó dret del ratolí a la imatge i seleccionar *Guardar imatge com...* (o l'opció equivalent al vostre navegador) per guardar-la al vostre equip (vegeu la figura 1.65).

FIGURA 1.65. Desar el GIF generat des del navegador



Cal tenir en compte, en aplicar filtres, que aquests no sempre tenen l'efecte desitjat i poden produir l'aparició d'artefactes a les imatges. Per exemple, en aplicar el filtre sèpia al GIF generat en aquest exemple el resultat ha estat el que es pot apreciar a la figura 1.66.

FIGURA 1.66. Resultat erroni amb artefactes



En canvi, aplicant el filtre *grayscale* la imatge es genera correctament.

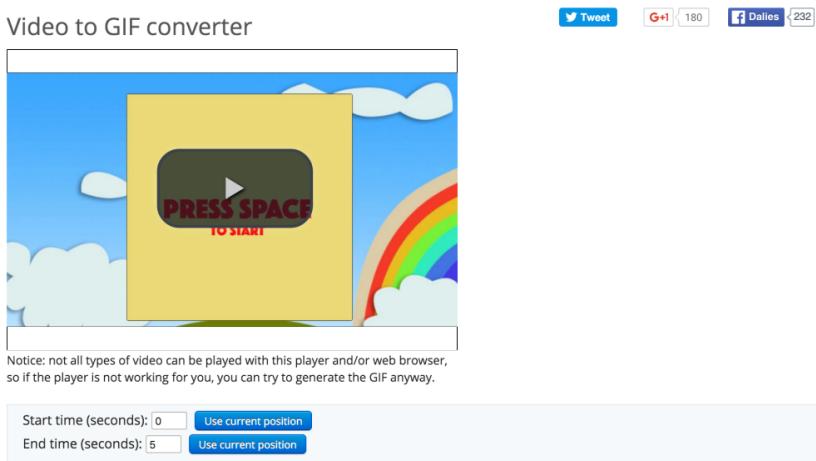
Creació d'un GIF animat a partir d'un vídeo

El primer pas és obtenir el vídeo que voleu convertir, i una vegada el tingueu heu d'assegurar-vos que el seu format és MP4, AVI o WEBM.

A continuació heu d'entrar a la web ezgif.com/video-to-gif i seguir aquests passos:

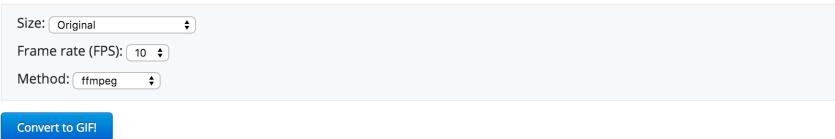
1. Trieu el vídeo que voleu carregar (formats MP4, AVI o WEBM, i màxim 60 MB).
2. Cliqueu al botó *Upload!*. El procés pot trigar força temps depenent de la velocitat de la vostra connexió, i no es mostra cap progressió.
3. Una vegada el fitxer estigui carregat podreu veure el vídeo original i establir el temps d'inici i fi que voleu incloure al vostre GIF com es pot apreciar a la figura 1.67.

FIGURA 1.67. Vídeo carregat correctament



4. Sota el vídeo podeu trobar les opcions per a la creació del GIF (vegeu figura 1.68); seleccioneu la mida que vulgueu, el nombre de *frames* per segon i el mètode (no cal canviar-lo). Cliqueu sobre el botó *Convert to GIF* per portar a terme la conversió.

FIGURA 1.68. Opcions per generar el GIF a partir d'un fitxer de vídeo

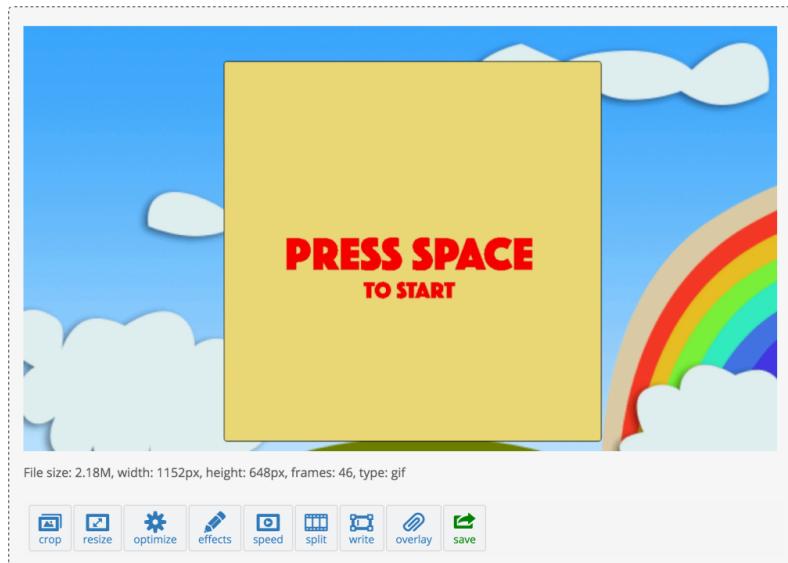


5. Com a l'exemple anterior, a la part inferior apareixerà la barra d'eines, que us permet aplicar diferents modificacions al nostre GIF. Per acabar, només l'heu de guardar al vostre equip (vegeu la figura 1.69).

FIGURA 1.69. Opcions d'edició del GIF

Output gif:

(Please be patient, video conversion may take a while, especially for long gifs with high frame rate)

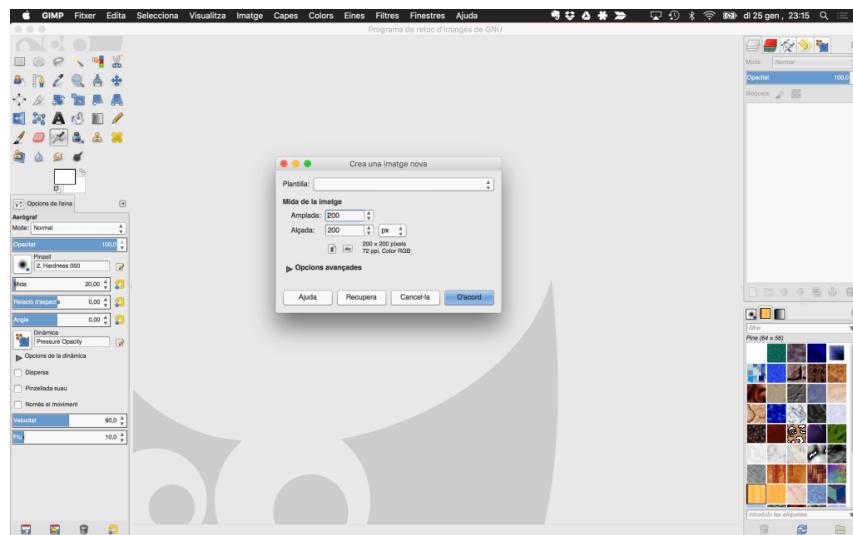


Una altra opció és fer servir un programari d'edició d'imatges, com Adobe Photoshop o GIMP. En general, aquest tipus de programari ens permet exportar directament la imatge com a GIF animat, fent servir les diferents capes per extreure la composició de cada *frame*.

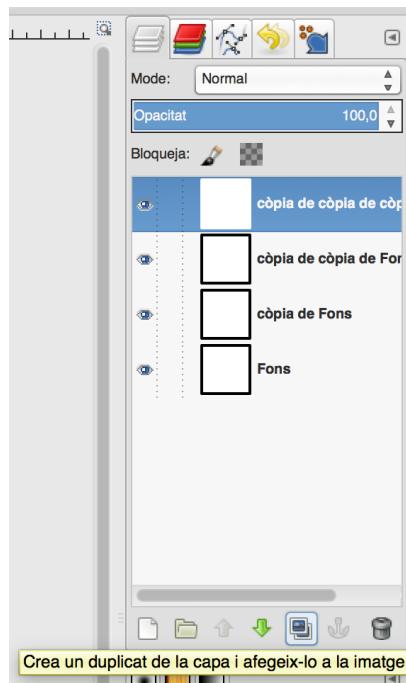
Creació d'un GIF animat a partir d'una seqüència d'imatges amb GIMP

Per crear un GIF animat amb GIMP heu de seguir els següents passos:

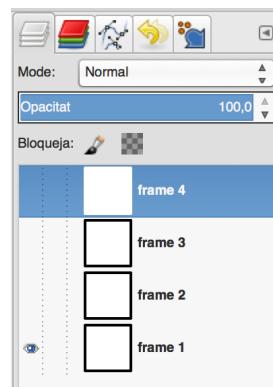
1. Obriu GIMP i creeu una nova imatge amb la mida que vulgueu per al vostre GIF animat (vegeu la figura 1.69).

FIGURA 1.70. Creació d'una imatge nova amb GIMP

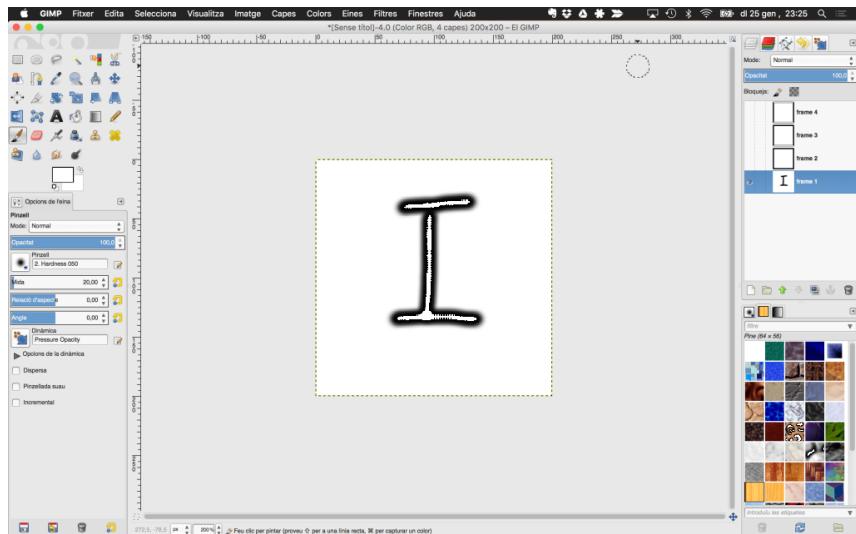
2. Seleccioneu la capa que s'ha creat automàticament i premeu el botó *Duplicar capa* tantes vegades com *frames* vulgueu a la vostra animació (vegeu la figura 1.71).

FIGURA 1.71. Selecció de capes

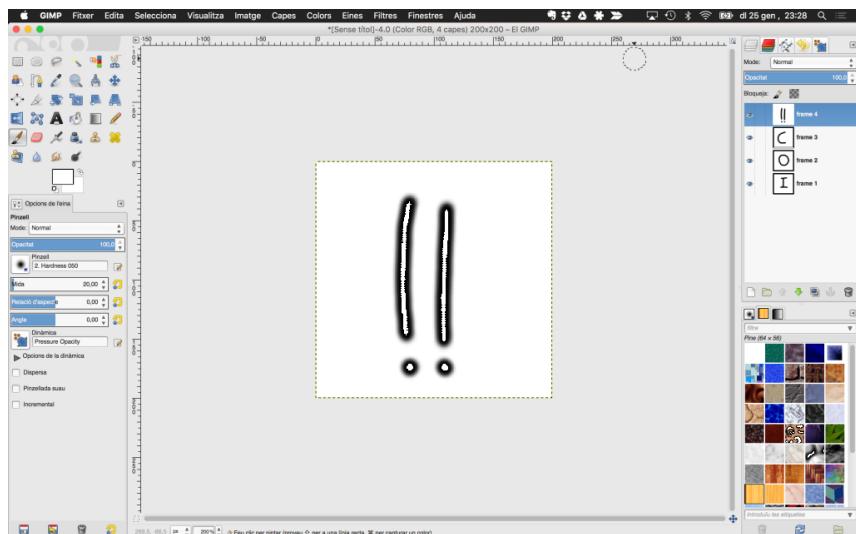
3. Canvieu el nom de les capes, fent doble clic sobre el nom, per millorar l'*organització*. El primer *frame* es corresindrà amb la capa inferior i l'últim, amb la capa superior. Feu clic sobre la icona de l'ull que hi ha al costat de cada capa excepte la inferior per poder veure els canvis que hi fareu (vegeu la figura 1.72).

FIGURA 1.72. Organitzacio de capes fent servir noms

4. Amb la primera capa seleccionada (que ha de ser l'única visible), dibuixe el contingut del primer *frame* (vegeu la figura 1.73).

FIGURA 1.73. Contingut del frame 1

5. Repetiu el procés amb la resta de capes fins a tenir-les totes dibuixades (vegeu la figura 1.74).

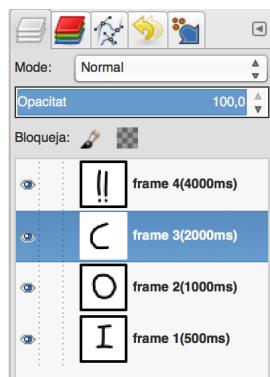
FIGURA 1.74. Contingut de l'últim frame

6. Una vegada tingueu totes les imatges aneu a *Filtres / Animació / Reprodueix...*. Apareixerà la finestra de reproducció, on podreu ajustar els frames per segon i la velocitat de reproducció (vegeu la figura 1.75).

FIGURA 1.75. Reproductor d'animacions de GIMP

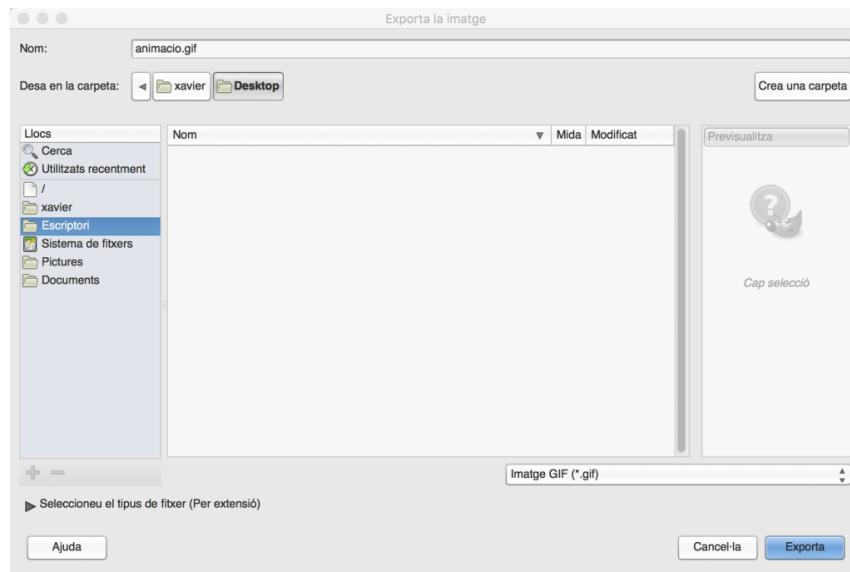
7. Es pot ajustar la durada de cada *frame* especificant en el nom de la capa la durada en ms, com es mostra a la figura 1.76.

FIGURA 1.76. Durada de cada frame ajustada a través del nom de la capa



8. Una vegada tingueu la durada ajustada, l'últim pas és exportar l'animació com a GIF animat. Aneu a *Fitxer / Exporta*. A la finestra heu de posar el nom que voleu donar al vostre fitxer, indiqueu on el voleu guardar i seleccioneu com a tipus de fitxer **imatge GIF (*.gif)** (vegeu la figura 1.77).

FIGURA 1.77. Exportació d'imatges amb GIMP



9. A les opcions per exportar la imatge com a GIF marqueu la casella *Com una animació* i cliqueu al botó *Exporta* (vegeu la figura 1.78).

FIGURA 1.78. Opcions d'exportació com a GIF

El GIF animat es trobarà a la ubicació seleccionada i podreu visualitzar l'animació amb qualsevol navegador web.

Canvas

A HTML5 es va incorporar un nou element anomenat **canvas**. La funció d'aquest element és força diferent de la de la resta d'elements que es troben en una pàgina web.

Per si mateix no fa res, i les úniques propietats que podem establir són l'amplada i l'alçada.

```

1 <canvas id="canvas" width="300" height="300">
2   El teu navegador no suporta l'element canvas
3 </canvas>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/NxLggX.

El **canvas** és com un tapís dintre del navegador que permet dibuixar-hi, escriure-hi o afegir-hi imatges.

```

1 <style>
2   canvas{
3     border: 1px solid black;
4   }
5 </style>
6
7 <script>
8   var canvas = document.getElementById("canvas");
9   var ctx = canvas.getContext("2d");
10
11   ctx.fillStyle = "#2897E8";
12   ctx.fillRect(10, 10, 100, 100);
13 </script>
14
15 <canvas id="canvas" width="200" height="200">
16   El teu navegador no suporta l'element canvas
17 </canvas>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/MKqooP.

Si us fixeu en l'exemple, primer s'obté una referència a l'element canvas, i a continuació el seu context. És sobre aquest context sobre el qual s'apliquen totes les accions, com per exemple canviar l'estil per dibuixar amb el color blau clar, `fillStyle()`, o pintar un quadre, `fillRect()`.

A banda de rectangles, hi ha una altra forma geomètrica que es pot crear al canvas: les rutes (*path* en anglès).

Es pot trobar un tutorial complet de l'API en el següent enllaç:
goo.gl/xWBfG9.

```

1 <script>
2   var ctx = canvas.getContext('2d');
3
4   ctx.fillStyle = "#2897E8"
5   ctx.beginPath();
6   ctx.moveTo(75, 50);
7   ctx.lineTo(100, 75);
8   ctx.lineTo(100, 25);
9   ctx.fill();
10 </script>
11 <code html>
12 <canvas id="canvas" width="300" height="300">
13   El teu navegador no suporta l'element canvas
14 </canvas>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/OMogwO.

També és possible dibuixar text dins del canvas, com es pot veure en el següent exemple:

```

1 <script>
2   var ctx = canvas.getContext('2d');
3
4   ctx.font = "30px Times New Roman";
5   ctx.fillStyle = "Black";
6   ctx.fillText("IOC", 5, 30);
7   ctx.strokeText("IOC", 60, 30);
8 </script>
9 <canvas id="canvas" width="300" height="300">
10   El teu navegador no suporta l'element canvas
11 </canvas>
```

Podeu veure aquest exemple en funcionament en el següent enllaç: codepen.io/ioc-daw-m09/pen/gPdRZG.

TAULA 1.24. Resum de mètodes i propietats del context

Mètode	Descripció	Exemple
<code>fillStyle</code>	Estableix el color amb què es pintarà	<code>context.fillStyle = "#2897E8"</code>
<code>strokeStyle</code>	Estableix el color de les vores	<code>context.strokeStyle = "#000"</code>
<code>fillRect(x, y, width, height)</code>	Pinta un rectangle farcit	<code>context.fillRect(10,10,100,100)</code>
<code>strokeRect(x, y, width, height)</code>	Pinta un rectangle, només les vores	<code>context.strokeRect(10,10,100,100)</code>
<code>clearRect(x, y, width, height)</code>	Esborra el contingut del rectangle	<code>context.clearRect(10,10,100,100)</code>

TAULA 1.24 (continuació)

Mètode	Descripció	Exemple
<code>beginPath()</code>	Inicia una ruta	<code>context.beginPath()</code>
<code>moveTo(x,y)</code>	Mou el cursor intern sobre el canvas	<code>context.moveTo(75,50)</code>
<code>lineTo(x,y)</code>	Dibuixa una línia des del punt actual fins al punt indicat	<code>context.lineTo(50,75)</code>
<code>fill()</code>	Omple la ruta amb el color seleccionat	<code>context.fill()</code>
<code>font()</code>	Estableix la font per dibuixar text	<code>context.font = "20px Times New Roman"</code>
<code>fillText(text, x, y[, maxWidth])</code>	Omple el text a les coordenades indicades	<code>context.fillText("IOC", 5, 30);</code>
<code>strokeText(text, x, y[, maxWidth])</code>	Dibuixa les vores del text a les coordenades indicades	<code>context.strokeText("IOC", 5, 30);</code>

Per obtenir el context s'ha d'obtenir primer la referència al canvas i després cridar `getContext("2d")` sobre aquest element.

La taula 1.24 no és exhaustiva. L'API ens permet també dibuixar arcs, corbes, degradats, afegir-hi imatges i crear patrons a partir d'aquestes.

Combinant els dibuixos al tapís amb les crides a `window.requestAnimationFrame()` es poden crear animacions; així és com es fan les animacions per a jocs i per a alguns bànders publicitaris que podem trobar a Internet.

Generalment es crida aquest mètode per repaintar l'escena, ja sigui esborrant-la primer i cridant el mètode del context `clearRect(x, y, width, height)`, o repaintant tot a sobre com es fa al joc IOC Invaders. El primer mètode és recomanable quan només canvien alguns sectors de la pantalla, i el segon és preferible quan canvia tot el contingut.

En el joc IOC Invaders es fa servir com a fons un *scroll* múltiple (desplaçament del fons) i, per tant, cal repaintar tot el tapís cada vegada; en canvi, a IOC Puzzle només cal repaintar la posició de les peces que s'hagin mogut i és més eficient que el primer mètode.

Podeu donar una ullada als següents exemples, encara que són força complexos, per comprovar les possibilitats que ens ofereix el canvas:

- Demostració de les capacitats de dibuix sobre canvas fent servir JavaScript: codepen.io/ioc-daw-m09/pen/LGJEyj.
- Un altre *pen* mostrant les possibilitats del canvas, aquest cop amb partícules: codepen.io/ioc-daw-m09/pen/adazjw.

Jocs HTML5 que fan servir el canvas

Podeu trobar exemples complets d'ús del canvas als jocs IOC Puzzle (github.com/XavierGaro/ioc-puzzle-lite) i IOC Invaders (github.com/XavierGaro/ioc-invaders).

2. Continguts interactius al web. Introducció a jQuery

Per desenvolupar un lloc web no tan sols cal preparar els elements multimèdia com són les imatges, el vídeo i el so, sinó que s'ha de ser capaç d'integrar-los. Per fer-ho, molt sovint són necessaris coneixements de JavaScript, a banda d'HTML i CSS.

JavaScript és l'únic llenguatge disponible nadiuament als navegadors i per aquesta raó és el que es fa servir per afegir comportaments dinàmics i interactius al web. Atesa la seva importància, és imprescindible adquirir uns coneixements bàsics sobre aquest.

Podeu trobar una guia bàsica de JavaScript en la secció "Annexos" del web del mòdul.

Habitualment, es fan servir biblioteques que faciliten molt la feina perquè incorporen dreceres per a les tasques més habituals o implementacions genèriques de controls que es poden fer servir en diferents projectes.

2.1 Elements interactius bàsics. Introducció a JQuery

La biblioteca jQuery és una de les més utilitzades i ofereix moltes possibilitats, la qual cosa fa que hi hagi una quantitat extensa de recursos de consulta, tutorials i documentació a Internet que podeu consultar per avançar més en aquest tema pel vostre compte.

Entre d'altres, jQuery facilita la modificació de l'estructura del document, afegir i modificar classes CSS i afegeix moltes funcions que faciliten moltes de les tasques més comunes en treballar amb la interfície del lloc web.

AJAX és un conjunt de tècniques de programació que permeten enviar peticions al servidor i rebre la resposta sense haver de recarregar la pàgina ni obrir-ne una de nova.

2.1.1 Configuració dels navegadors per a la visualització d'elements interactius

Abans de l'arribada d'HTML5 i les millores aportades per CSS3, quan es volien afegir elements interactius a un lloc web s'havia de recórrer a tecnologies alienes al navegador com Java (amb unes miniaplicacions anomenades *applets*) o molt més freqüentment a Flash. Aquestes tecnologies avui dia són obsoletes i han estat reemplaçades per l'ús d'HTML5, CSS i JavaScript.

Això simplifica molt la configuració dels navegadors, ja que no cal descarregar cap connector especial, com requeria Flash, ni la màquina virtual de Java. En ser tecnologies nadiues del navegador, poden fer-se servir directament.

Encara que no us heu de preocupar del fet que l'usuari hagi d'instal·lar cap connector en particular, sí que hi ha dues opcions que es poden modificar o desactivar que poden fer que el lloc web deixi de funcionar i mostrar-se correctament:

-
- Les *cookies* són petits fitxers que guarda el navegador amb informació.
- Desactivar l'ús de *cookies*: com que no es podrà *recordar* la informació de l'usuari, alguns llocs seran directament inaccessible, per exemple els llocs web de comerç electrònic.
 - Desactivar l'ús de JavaScript: en alguns casos el lloc serà inservible. Per exemple, una web que inclogui mapes de Google Maps no pot funcionar sense JavaScript; en canvi, d'altres només l'utilitzen per afegir alguns efectes, i per a aquests sí que es pot presentar una solució alternativa.

Aquests casos s'han de tenir en compte i, com a mínim, s'ha de mostrar un missatge avisant l'usuari que el lloc no pot funcionar o té limitades les seves funcionalitats, perquè aquestes opcions estan desactivades o no estan disponibles.

Per altra banda, amb la inclusió de nous dispositius d'entrada i de sortida s'han afegit alguns controls que limiten a quins d'aquests dispositius es pot accedir. Per exemple, les següents API demanden permís a l'usuari abans de permetre utilitzar-les:

- l'API de geolocalització
- l'API de càmera
- l'API del micròfon

Aquests permisos s'han d'atorgar individualment per a cada pàgina, de manera que, per exemple, un lloc maliciós no pot fer servir la *webcam* sense el consentiment previ.

Tot i així, s'ha de tenir en compte que no tots els navegadors suporten totes les característiques previstes per HTML5, sobretot si es consideren les Web API que van afegint-se contínuament o si voleu fer servir alguna API o biblioteca experimental, com per exemple la WebVR API, que encara està disponible només com a esborrany i només és suportada per Google Chrome i les versions més recents de Mozilla Firefox.

Un altre exemple més habitual és oferir un mètode alternatiu per reproduir àudio i vídeos quan el navegador no suporta HTML5. Actualment no hi ha cap solució suportada universalment, així que l'únic que es pot fer és mostrar un missatge a l'usuari:

```
1 <video>
2   <source src="fitxer_video.mp4" type="video/mp4">
3   <source src="fitxer_video.webm" type="video/webm">
4   Ho sento, el teu navegador no suporta vídeo d'HTML5
5 </video>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/xZjRVN.

Podeu trobar més informació sobre la WebVR API en el següent enllaç: goo.gl/4NP6xB.

Com es pot veure en aquest exemple:

1. Primerament, s'intenta fer servir com a font del vídeo un fitxer en format MP4.
2. Si aquest format no és reconegut, s'intenta amb el format WEBM.
3. Si cap dels dos és reconegut, es mostra un missatge informant l'usuari.

Aquests tipus de situacions es poden trobar sempre que es treballa amb HTML5, CSS3 i JavaScript. Poden afrontar-se des de dos angles diferents:

- **Progressive enhancement:** es comença a desenvolupar el lloc amb les funcionalitats bàsiques que no requereixen cap tecnologia i es va millorant l'experiència d'usuari afegint nous comportaments (i comprovant-los) pas a pas.
- **Graceful degradation:** amb aquesta tècnica es treballa de manera inversa, primer es desenvolupa el lloc amb totes les tecnologies que es necessiten i després s'afegeixen alternatives de visualització/execució per a les parts que no siguin disponibles globalment.

Podeu trobar més informació sobre les diferències entre *progressive enhancement* i *graceful degradation* en aquest enllaç: goo.gl/89UPNm.

Es pot **desactivar JavaScript** des de les opcions (pestanya *General* a Google Chrome) de les eines de desenvolupador, no cal canviar les preferències del navegador.

Vegeu un exemple de totes dues aproximacions. Suposeu que teniu una pàgina amb un botó que crida un *script* que fa uns càlculs:

```

1 <script>
2 function calcula() {
3     var a = 12,
4         b = 30,
5         resultat = a + b;
6
7     alert("Resultat: " + resultat);
8 }
9 </script>
10
11 <button onclick="calcula()">Calcular</button>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/ONPvgP.

Com que no hi ha manera de fer el càlcul si no es troba activat el JavaScript al navegador, heu d'afegir una implementació alternativa. En cas contrari, es produeix una mala experiència d'usuari per als visitants del lloc web que no el tinguin activat; per exemple, alguns usuaris amb dispositius mòbils.

Aplicant la tècnica de la *graceful degradation* podríeu fer el següent:

```

1 <script>
2   function calcula() {
3     var a = 12,
4       b = 30,
5       resultat = a + b;
6
7     alert("Resultat: " + resultat);
8   }
9 </script>
10
11 <button onclick="calcula();">Calcular</button>
12 <noscript>
13   <p>
14     Per realitzar el càlcul és necessari tenir JavaScript activat. Si us plau,
15       activeu-lo al vostre navegador.
16 </p>
17 </noscript>

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/RaNMxz (malauradament, si desactiveu el JavaScript al vostre navegador CodePen deixarà de funcionar, ho heu de provar en local).

Com podeu veure, s'ha fet servir l'element `noscript` per mostrar el missatge en cas que el navegador no tingui el JavaScript activat.

Tot i que ara es preveuen totes dues possibilitats, en el moment de la implementació s'ha fet sabent que és possible que alguns usuaris no tinguin activat JavaScript, i que fins i tot pot ser que no tinguin opció d'activar-lo, de manera que aquests usuaris es troben amb una pàgina amb un botó que no fa res i un missatge que tampoc no els ajuda a solucionar el seu problema.

En aquest cas en concret no seria possible aplicar el ***progressive enhancement*** sense recórrer a fer servir algun llenguatge al servidor, per exemple PHP, fer l'enviament de les dades a la pàgina i que aquesta retornés el resultat.

Un altre cas molt habitual en què és interessant aplicar la ***graceful degradation*** és quan es volen aplicar efectes CSS més avançats, per exemple un degradat:

```

1 <style>
2   div {
3     width: 100px;
4     height: 100px;
5
6     /* Degradat */
7     background: #f0f9ff; /* Navegadors antics, color pla */
8     background: -moz-linear-gradient(-45deg, #f0f9ff 0%, #cbebff 47%, #aldbff
9         100%); /* FF3.6-15 */
10    background: -webkit-linear-gradient(-45deg, #f0f9ff 0%, #cbebff 47%, #aldbff
11        100%); /* Chrome10-25, Safari5.1-6 */
12    background: linear-gradient(135deg, #f0f9ff 0%, #cbebff 47%, #aldbff 100%);
13      /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
14    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#f0f9ff',
15      endColorstr='#aldbff', GradientType=1 ); /* IE6-9 fallback on
16          horizontal gradient */
17  }
18 </style>
19 <div></div>

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/RaNEvV.

Hi ha moltes eines en línia que permeten generar automàticament el codi CSS aplicant la tècnica del ***graceful degradation***, per exemple:

[www.colorzilla.com/gradient-editor.](http://www.colorzilla.com/gradient-editor/)

Com que els fulls CSS s'apliquen en cascada, la primera declaració és sobreescrita per les posteriors, de manera que l'última en aplicar-se és l'última que sigui suportada. Així doncs, en els navegadors més antics s'aplicarà un color pla, en versions antigues dels navegadors que ho suporten es crida afegint el prefix (era necessari en versions anteriors), i els navegadors moderns fan servir la declaració de l'especificació, com indica el W3C.

A continuació veureu un exemple en què aplicar la tècnica del *progressive enhancement* sí que és una molt bona alternativa. Suposeu que teniu una versió més avançada de l'exemple anterior que permet introduir dos valors i mostrar el resultat.

Es comença afegint només l'estructura del formulari amb el marcat d'HTML:

```

1 <form action="" method="GET" id="principal">
2   <fieldset>
3     <legend>Calcula la suma de dos nombres</legend>
4     <label>Nombre A</label>
5     <input name="number_a" type="text">
6     <br>
7     <label>Nombre B</label>
8     <input name="number_b" type="text">
9     <br>
10    <input value="calcular" type="submit">
11  </fieldset>
12 </form>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/pyvLGW. En aquest cas en tractar-se d'un exemple no cal indicar l'URL a la que s'enviarà el formulari a través de la propietat `action`.

Com que només inclou HTML bàsic, podeu estar segurs que aquest exemple funcionarà en tots els navegadors (vegeu la figura 2.1).

FIGURA 2.1. Implementació simple amb HTML

S'afegeix una segona capa, aquest cop amb el codi CSS:

```

1 <style>
2   fieldset {
3     max-width: 300px;
4     border: 1px dotted grey;
5   }
6
7   legend {
8     text-transform: uppercase;
9     font-weight: italic;
10    font-size: 0.8em;
11  }
12
13  label {
14    font-weight: bold;
15    font-size: 1.1em;
16  }
```

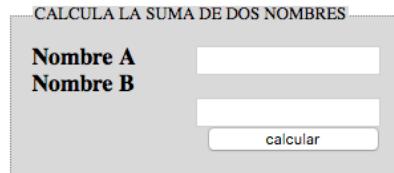
En alguns navegadors és possible desactivar el CSS, i en els casos dels lectors de pantalles utilitzats per les persones amb discapacitats visuals, els fulls CSS no són aplicables.

```

17      input {
18        width:50%;
19        float: right;
20      }
21    
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/VaYXRo i la seva visualització a la figura 2.2.

FIGURA 2.2. Implementació afegint CSS



Es conserva tota la funcionalitat, però ara s'ha millorat l'aspecte bàsic del formulari. El següent pas és afegir una nova capa amb codi JavaScript, que serà l'encarregat de validar la informació del formulari abans de fer l'enviament:

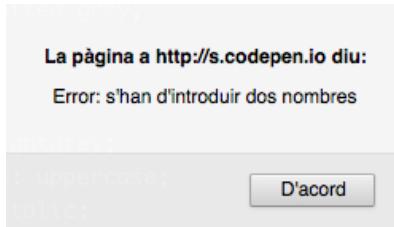
```

1 <script>
2   var formulari = document.getElementById('principal'),
3     numberA = document.getElementsByName('number_a')[0],
4     numberB = document.getElementsByName('number_b')[0];
5
6   formulari.addEventListener("submit", function(e) {
7
8     if (!isNumeric(numberA.value) || !isNumeric(numberB.value)) {
9       e.preventDefault();
10      alert("Error: s'han d'introduir dos nombres");
11    } else {
12      alert("Correcte: enviant formulari");
13    }
14  });
15
16  function isNumeric(n) {
17    return !isNaN(parseFloat(n)) && isFinite(n);
18  }
19 </script>

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/ONPadN i la seva visualització a la figura 2.3.

FIGURA 2.3. Implementació afegint JavaScript



Aplicant el **progressive enhancement** s'ha fet que el codi sigui funcional per a tots els usuaris, millorant l'experiència d'usuari a mesura que les capacitats dels navegadors dels usuaris augmenten.

2.1.2 Eines per a la inclusió de contingut multimèdia

Com que actualment tot el contingut multimèdia que es troba al web és gestionat i manipulat fent servir JavaScript, HTML i CSS, no hi ha gaires eines que alleugereixin la nostra tasca.

Existeixen eines que exporten continguts com a HTML, CSS i JavaScript. Principalment es tractarà de *software* privat, com Adobe Animate CC, que permet la creació de continguts multimèdia i que exporta aquests com a codi HTML, CSS i JavaScript. O eines com Unity 3D, que permet desenvolupar jocs fent servir el llenguatge C# i exportar-los a diferents plataformes, entre les quals HTML5 i WebGL. Però com a desenvolupadors serà més habitual que us trobeu treballant amb codi.

El que sí que utilitzareu molt sovint són biblioteques que us facilitaran enormement la vostra tasca, ja que automatitzen les accions més repetitives, simplifiquen la sintaxi i s'encarreguen de gestionar les diferències entre navegadors, fent servir internament diferents implementacions segons siguin o no suportades determinades característiques.

Un altre tipus d'eina molt útil són les aplicacions en línia que es poden trobar per resoldre problemes molt concrets i que us generen codi automàticament, per exemple els generadors de codi CSS3 que faciliten la tasca d'afegirombres, degradats i altres efectes a partir d'una interfície gràfica. Al llarg d'aquest apartat utilitzareu algunes d'aquestes eines per ajudar-vos a desenvolupar el vostre codi.

Si accediu a la pàgina de descàrrega de jQuery (jquery.com/download) veureu que hi ha diverses opcions.

Per una banda es troba la versió anomenada *compressed* (comprimida) i per altra la versió *uncompressed* (descomprimida). Quan estigueu desenvolupant sempre heu d'optar per la versió sense comprimir, ja que si heu de depurar o veure com funciona internament alguna part de la biblioteca ho podreu fer sense problema. La versió comprimida és inintel·ligible per depurar, però té un pes menor i és la que s'ha de fer servir en l'entorn de producció.

Els **fitxers de JavaScript**, en ser text pla, realment no es comprimeixen, el que es fa és modificar-los fent servir altres eines de manera que s'eliminin tots els comentaris, els espais innecessaris i els salts de línia, i se substitueix el nom de les variables per d'altres més curts (normalment, una sola lletra). D'aquesta manera, la mida del fitxer es redueix i el temps de descàrrega és menor.

Una vegada decidíu quina versió voleu utilitzar només l'heu de descarregar i copiar al mateix directori que la resta de fitxers del nostre lloc web o en algun dels seus subdirectoris.

Per exemple, si heu descarregat la versió descomprimida, tindreu un fitxer amb un nom semblant a jquery-3.1.1.js. Suposeu que el copieu dins del subdirector

Podeu trobar més informació sobre jQuery a la seva pàgina oficial: jquery.com.

Es poden trobar versions les versions 1.x (suport per navegadors antics) i 2.x al següent enllaç: code.jquery.com

Compressors en línia

Una de les solucions més simples per comprimir els nostres fitxers CSS i JavaScript és fer servir un **compressor en línia**, com per exemple jscompress.com, que permet comprimir el codi JavaScript només enganxant-lo i clicant un botó.

js/biblioteques dins del directori que conté tots els vostres fitxers. Per carregar-lo a la pàgina hauríeu d'afegir:

¹ `<script src="js/biblioteques/jquery-3.1.1.js"></script>`

Xarxes de lliurament de continguts (CDN)

Un CDN és una xarxa de servidors localitzats en diferents punts geogràfics però amb els mateixos continguts, de manera que quan es rep una petició del fitxer aquest és enviat des del servidor més proper a l'usuari. D'aquesta manera, s'augmenta la velocitat de la resposta. Podeu trobar més informació sobre les xarxes de lliurament de continguts en el següent enllaç: ca.wikipedia.org/wiki/Xarxa_de_lliurament_de_continguts.

En alguns casos pot interessar-vos no allotjar la biblioteca en el vostre servidor. Per exemple, si la pàgina web es preveu que sigui visitada per usuaris internacionals, serà una millor opció fer servir un CDN (xarxa de lliurament de continguts); en el cas de jQuery, ells mateixos ens ofereixen aquest servei, i és tan simple d'utilitzar com afegir la ruta `code.jquery.com/jquery-3.1.1.js` (substituint el nom del fitxer per la versió que correspongui) en lloc de la del fitxer descarregat, per exemple:

¹ `<script src="//code.jquery.com/jquery-3.1.1.js"></script>`

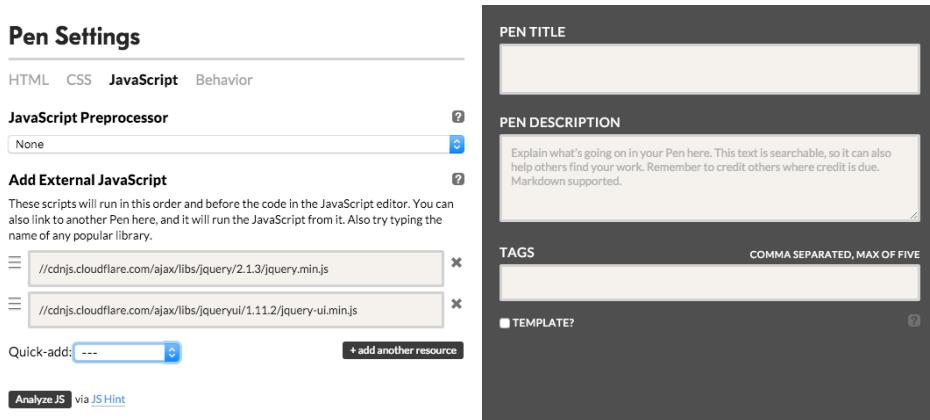
No es pot **carregar una biblioteca** fent servir un CDN si obrim la pàgina directament al nostre ordinador com un fitxer local. En aquests casos haureu de descarregar el fitxer i carregar-lo localment, o bé accedir a la pàgina a través d'un servidor web que tinguem configurat en el nostre equip.

2.1.3 Introducció a jQuery

El primer pas per poder fer servir jQuery al codi és carregar la biblioteca corresponent. En els exemples que es veuran a continuació es fa servir la versió 3.1.1 i és càrrega a través CDN proporcionat pels desenvolupadors de jQuery:

¹ `<script src="//code.jquery.com/jquery-3.1.1.js"></script>`

CodePen inclou l'opció de carregar automàticament algunes de les biblioteques externes més populars (vegeu la figura 2.4), com és el cas de JQuery i JQuery UI. Per aquesta raó no s'inclou la càrrega d'aquestes al codi d'exemple, perquè són carregades automàticament.

FIGURA 2.4. Configuració de CodePen

S'han de tenir en compte dues coses abans de treballar amb jQuery:

- S'ha de carregar la biblioteca abans de fer-la servir.
- El **DOM** (l'estructura del document) ha d'estar completament carregat.

El model d'objectes del document, més conegut com a **DOM** (*Document Object Model*) per les seves sigles en anglès, és una interfície que ens facilita treballar amb documents HTML, entre d'altres.

El DOM defineix els mètodes i les propietats que ens permeten accedir i manipular el document com si es tractés d'un arbre de nodes, on l'arrel és el node document que pot contenir qualsevol quantitat de nodes fills i les fulles, els nodes que no tinguin cap descendent (generalment, el contingut dels elements HTML).

Una de les funcionalitats que inclou jQuery és la capacitat de seleccionar i manipular aquests elements fàcilment, ja sigui per modificar-los, per eliminar-los o per afegir-ne de nous.

Encara que el primer punt sembla obvi, com que és possible carregar els fitxers amb el codi de manera asíncrona és molt probable que el vostre fitxer amb el codi JavaScript acabi de descarregar-se abans que la biblioteca, pel fet de ser més lleuger.

El segon punt pot fer que apareguin problemes de difícil detecció, ja que pot ser que comenci a executar-se el codi abans que s'hagi generat el DOM completament.

Per aquesta raó, sempre s'ha d'afegir el vostre codi dins de la funció que és cridada pel mètode `ready` de jQuery, que és cridat una vegada s'ha generat el DOM completament:

```

1 <script src="//code.jquery.com/jquery-3.1.1.js"></script>
2 <script>
3   $(document).ready(function() {
4     // Aquí va el nostre codi
5   });
6 </script>

```

Generalment, les biblioteques fan servir variables i mètodes amb aquests símbols per ressaltar que es tracta d'alguna funció, propietat o objecte especial.

Hi ha casos en què cal utilitzar jQuery com a nom per evitar conflictes amb altres biblioteques.

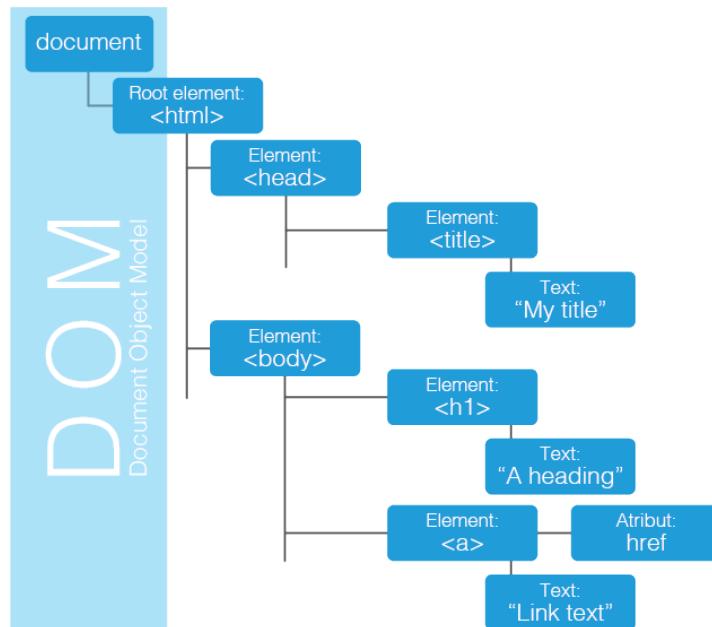
Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/KzwEXN.

El primer que haureu vist és que hem fet servir el símbol del dòlar (\$). A JavaScript es pot fer servir com a nom o part del nom de les variables i funcions. És a dir, **és possible, però no recomanable**, posar com a nom a una variable \$\$variable\$\$.

En aquest cas, el símbol \$ és una drecera per a jQuery, així que \$(document) el que fa és cridar la funció jQuery(document).

A continuació, cal tenir clar què és aquest paràmetre que s'ha passat a la funció jQuery. Quan una pàgina web és carregada en un navegador, el codi es converteix en una estructura de dades similar a un arbre: el DOM. El node arrel d'aquest arbre és l'anomenat document, i la resta d'elements pengen d'aquest dins d'aquesta estructura. Podeu veure una representació d'aquest arbre a la figura 2.5.

FIGURA 2.5. Representació del DOM per a un document HTML



Així doncs, quan es fa servir el mètode ready sempre passarem l'objecte document com a paràmetre, ja que aquest és el responsable d'indicar quan s'ha acabat de generar l'arbre perquè tots els elements de la pàgina són descendents seus.

En síntesi, aquest exemple el que fa és *dir-li* a jQuery (\$) que es vol executar el codi (function() {};), una vegada el document (document) estigui llest (ready).

El mètode `ready()`, encara que té una funció similar a `<body onload="funcioACridar();">`, és **incompatible amb aquest**. Si es fa servir un no s'ha de fer servir l'altre.

S'han de distingir dues parts en aquest codi. Per una banda, es fa una crida a la funció jQuery, que retorna un objecte, i després es crida un mètode d'aquest objecte (en aquest cas es tractava de `ready`).

La funció jQuery converteix el paràmetre en un objecte jQuery, i aquest paràmetre pot ser un node com `document`, una cadena de text amb un **selector CSS**, com per exemple `p`, o una cadena de codi HTML, com per exemple `<p>Hola, món</p>`.

Vegem aquests objectes amb el següent exemple:

```

1 <script src="//code.jquery.com/jquery-3.1.1.js"></script>
2 <script>
3 $(document).ready(function() {
4   console.log('document:', $(document));
5   console.log('p: ', $('p'));
6   console.log('<p>Hola, Món</p>: ', ('<p>Hola, Món</p>'));
7 });
8 </script>
9 <p></p>

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/GZJJzY.

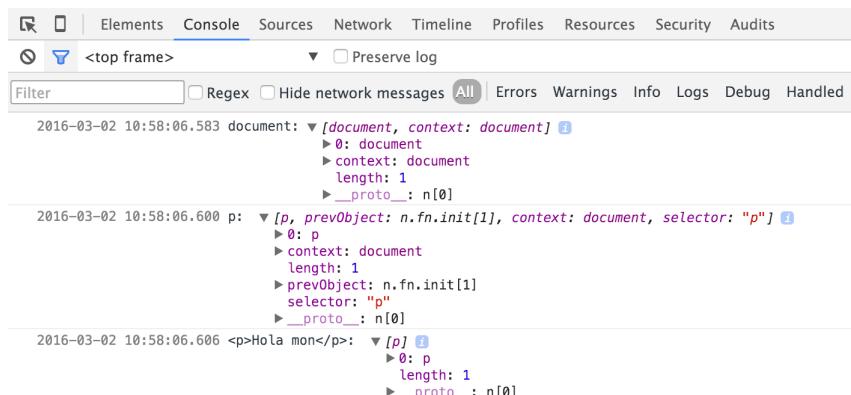
El mètode "console.log"

El mètode `console.log` mostra els valors dels paràmetres per a la consola de les eines de desenvolupador. Per poder veure el resultat heu d'anar a les opcions del vostre navegador:

- Google Chrome: botó desplegable d'*Opcions del navegador / Més eines / Eines de desenvolupador*.
- Mozilla Firefox: menú *Eines / Desenvolupador web / Mostra les eines*.

Una vegada es mostri el panell d'eines, veureu una pestanya anomenada *Console* o *Consola*; si la seleccioneu, podreu veure el resultat, similar a la figura 2.6.

FIGURA 2.6. Visualització de la consola de Google Chrome



The screenshot shows the Google Chrome DevTools interface with the 'Console' tab selected. The top navigation bar includes 'Elements', 'Console', 'Sources', 'Network', 'Timeline', 'Profiles', 'Resources', 'Security', and 'Audits'. Below the tabs, there's a dropdown for 'frame' set to '<top frame>' and a checkbox for 'Preserve log'. A 'Filter' input field is present. The main area displays three log entries from March 2, 2016, at 10:58:06.583, 10:58:06.600, and 10:58:06.606. Each entry shows the context object (document or p) and its properties like length and __proto__.

```

2016-03-02 10:58:06.583 document: ▼ [document, context: document] ⓘ
  ► 0: document
  ► context: document
  length: 1
  ► __proto__: n[0]

2016-03-02 10:58:06.600 p: ▼ [p, prevObject: n.fn.init[1], context: document, selector: "p"] ⓘ
  ► 0: p
  ► context: document
  length: 1
  ► prevObject: n.fn.init[1]
  selector: "#"
  ► __proto__: n[0]

2016-03-02 10:58:06.606 <p>Hola món</p>: ▼ [p] ⓘ
  ► 0: p
  length: 1
  ► __proto__: n[0]

```

Si desplegueu els continguts de cada objecte podreu veure que la seva estructura és diferent. Cal destacar que:

- L'objecte creat pel document no té la propietat `prevObject` perquè no hi ha cap objecte anterior, ni la propietat `selector`.
- L'objecte generat a partir del selector `p` conté tant un objecte anterior (`prevObject`), que es correspon amb el document (és lògic, perquè l'element està inclòs en el document), i la propietat `selector`.
- L'objecte generat a partir de la cadena de text no té cap element previ perquè no s'ha afegit al document; tampoc no té context, per la mateixa raó, i no es pot seleccionar.

Tots tenen la propietat `length` amb valor 1, això és perquè s'ha trobat només un element de cada tipus. Aquesta és una propietat important, perquè quan es treballa amb selectors l'objecte jQuery habitualment contindrà seleccionats múltiples elements:

```

1 <script src="//code.jquery.com/jquery-3.1.1.js"></script>
2 <script>
3   $(document).ready(function() {
4     console.log('p: ', $('p'));
5   });
6 </script>
7 <div>
8   <p>Primer paràgraf</p>
9   <p>Segon paràgraf</p>
10  <p>Tercer paràgraf</p>
11 </div>
12 <p>Quart paràgraf</p>
```

Selectors jQuery

jQuery suporta tots els selectors de CSS3, a més de XPath i alguns propis, com es pot veure en el següent enllaç: goo.gl/vIHEBN.

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/GZJpWL.

En aquest exemple podeu veure que la funció ha retornat un objecte amb mida 4 perquè s'han trobat quatre elements que coincideixen amb el selector `p` que correspon al paràgraf, com es pot veure a la figura 2.7.

FIGURA 2.7. Selecció d'elements de tipus paràgraf

The screenshot shows the CodePen interface with three main sections: HTML, CSS, and JS. The HTML section contains the following code:

```
<div>
  <p>Primer paràgraf</p>
  <p>Segon paràgraf</p>
  <p>Tercer paràgraf</p>
  <p>Quart paràgraf</p>
```

The CSS section is empty. The JS section contains:

```
$(document).ready(function() {
  console.log('p: ', $('p'));
});
```

At the bottom, the console tab shows the output of the console.log statement:

```
[p, p, p, p, prevObject: n.fn.init[1], context: document, selector: "p"]
```

En canvi, si canvieu el selector per `div > p`, el resultat és diferent, ja que només són seleccionats els paràgrafs que s'hi troben directament dins d'un element `div`:

div. Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/grpaxN.

Per descomptat, poden seleccionar-se també elements per a id, només cal afegir el símbol # a l'identificador:

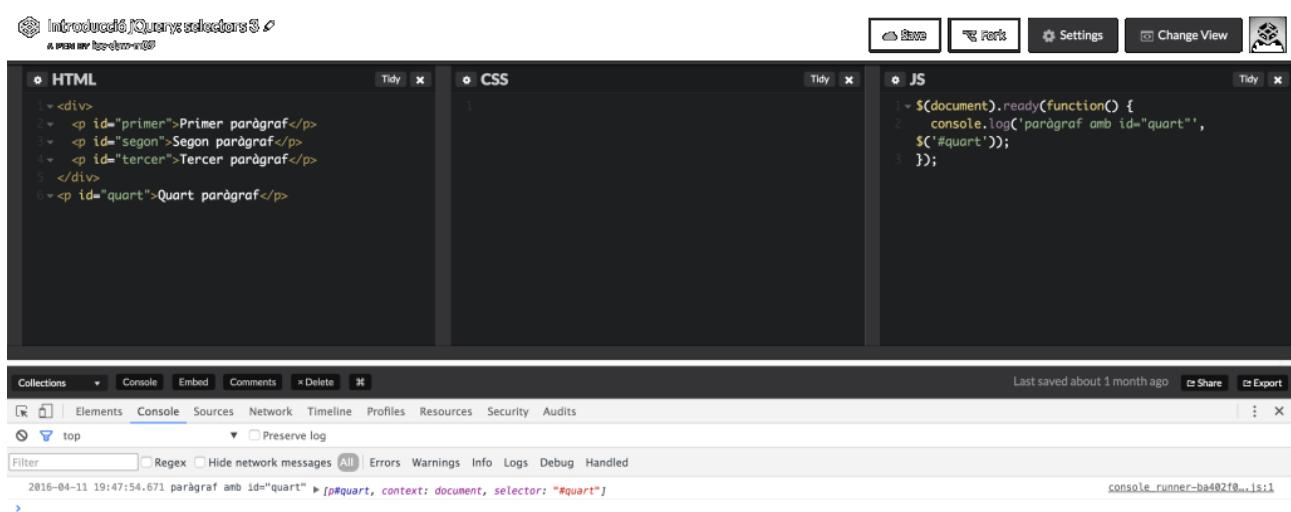
```

1 <script>
2   $(document).ready(function() {
3     console.log('paràgraf amb id="quart"', $('#quart'));
4   });
5 </script>
6 <div>
7   <p id="primer">Primer paràgraf</p>
8   <p id="segon">Segon paràgraf</p>
9   <p id="tercer">Tercer paràgraf</p>
10 </div>
11 <p id="quart">Quart paràgraf</p>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/GZJmaY.

A la consola aquest cop la mida és 1, i només conté l'element amb id="quart", com es pot veure a la figura 2.8.

FIGURA 2.8. Selecció d'un element per la seva id



Ara que ja heu vist com seleccionar un o més elements de la pàgina, vegeu com podeu aplicar un mateix canvi a tots aquests, per exemple per modificar el seu estil CSS:

```

1 <style>
2 div {
3   width: 100px;
4   height: 100px;
5   margin: 10px;
6   float: left;
7   background-color: lightgrey;
8 }
9 </style>
10 <script>
11 $(document).ready(function() {
12   var $quadres = $('div');
13   $quadres.css('background-color', 'red');
14   alert("El color dels quadres és: " + $quadres.css('background-color'));
15 });
```

```

16  </script>
17  <div id="primer"></div>
18  <div id="segon"></div>
19  <div id="tercer"></div>
20  <div id="quart"></div>

```

Quan es fa servir una variable que emmagatzema un objecte jQuery s'acostuma a prefixar amb el símbol \$, de manera que és molt clar que es tracta d'un objecte jQuery.

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/Mywowy.

Primerament, s'ha guardat el resultat de la funció en \$quadres, que contindrà l'objecte retornat per jQuery. A continuació s'ha cridat el mètode css() per establir la propietat background-color al color vermell, i finalment es mostra un quadre d'alerta recuperant el valor de la propietat background-color de l'objecte jQuery, com es pot apreciar a la figura 2.9.

FIGURA 2.9. Modificació del color mitjançant jQuery



Convé destacar que **es fa servir el mateix mètode tant per establir la propietat com per recuperar el seu valor**. jQuery distingeix entre una i una altra segons el nombre de paràmetres, de manera que si se'n passen dos intentarà establir-la, i si només se'n passa un intentarà recuperar-la. Molts dels mètodes de jQuery funcionen d'aquesta manera.

Però com ho faríeu per aplicar canvis a cada element individualment? Una opció seria afegir un id diferent a cada element i després declarar una variable amb un objecte jQuery per a cadascun, però això no seria gens eficient. Per a aquests casos, jQuery ens proporciona un mètode per recórrer a tots els elements i aplicar-hi els canvis de manera individual, substituint el codi JavaScript pel següent:

```

1$(document).ready(function() {
2  var $quadres = $('div');
3
4  $quadres.css('background-color', 'red');
5
6  $quadres.each(function(i) {
7    var alcada = 25 + i * 25;
8    $(this).css('height', alcada + 'px');
9  });
10 });

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/GZJEpP, i el resultat a la figura 2.10.

FIGURA 2.10. Quadres amb la mida modificada individualment



Diferents usos d'each a jQuery

jQuery té dos mètodes anomenats `each` que fan coses diferents. Si es crida a partir d'un objecte amb una selecció d'elements, el que fa és recórrer a aquests elements i aplicar la funció cridada a cada element. En canvi, si es crida directament sobre jQuery (`$.each(array, funcio)`), el que fa és recórrer a l'`array` i aplicar la funció a cada element d'aquest.

En aquest últim cas seria més adient dir que el que hem fet és cridar la **funció each** de la biblioteca per diferenciar-lo del primer cas, en què s'ha cridat el mètode `each` d'una instància d'un objecte concret.

S'ha afegit una crida al mètode `each` i s'ha passat com a paràmetre una funció. Aquesta funció és cridada per a cadascun dels elements de l'objecte jQuery, en aquest cas els quatre elements `div`. El valor del paràmetre `i` es correspondrà en cada cas amb l'índex de l'objecte (sent 0 el primer i 3 l'últim en aquest exemple).

Dins de la funció, el que es fa és calcular l'alçada de cada element amb la fórmula $25 + i * 25$; a continuació es crea un nou objecte jQuery que fa referència a aquest element amb `$(this)` (com que no es fa res més amb aquest no cal guardar-lo a cap variable), i es crida el mètode `css` com en el cas anterior, però aquest cop per a la propietat `height`.

Per a la propietat `height` s'ha d'afegir un tipus d'unitat a més de la mida, per exemple: `px` o `em`.

Fixeu-vos que per poder manipular cadascun d'aquests elements s'ha hagut de cridar de nou la funció jQuery passant l'element (`this` fa referència a l'element processat).

Però en alguns casos pot interessar-vos aturar el recorregut, potser cerqueu un element en concret i una vegada el trobeu voleu sortir. En aquest cas només cal retornar `false` quan es doni la condició per finalitzar el recorregut; per exemple, afegint aquest codi dins de la funció cridada:

```

1 if (i === 1) {
2   return false;
3 }
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/MywoBo.

Aquest canvi fa que una vegada el valor de `i` sigui igual a 1 (just després de canviar la mida del segon element) es retorne `false` i els dos elements restants no siguin processats, i per tant la seva alçada es mantingui igual, com es pot veure a la figura 2.11.

FIGURA 2.11. Només la meitat dels quadres són processats



Fins ara heu vist com treballar amb tots els elements seleccionats, però com ho faríeu si només volguéssiu treballar amb un en concret? Hi ha dues maneres d'accedir a aquests elements: a través del mètode `get` o directament com si es tractés d'un *array* especificant l'índex. Substituïu el contingut del bloc de JavaScript pel següent:

```

1  $(document).ready(function() {
2      var $quadres = $('div');
3
4      $($quadres[2]).css('background-color', 'red');
5
6      $($quadres.get(0)).css('background-color', 'green');
7
8  });

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/KzpqGa, i el resultat a la figura 2.12.

FIGURA 2.12. Elements modificats mitjançant el seu índex



S'ha de tenir en compte que encara que és possible, **no és recomanable accedir a l'element com si es tractés d'un array**. S'ha de fer servir el mètode `get` perquè és el que la interfície de jQuery ens garanteix que funcionarà correctament.

En el primer cas s'ha seleccionat el tercer element com si es tractés d'un *array*. Fixeu-vos que per poder cridar el mètode `css` s'ha hagut de tornar a cridar la funció jQuery passant-li com a argument el node emmagatzemat a la tercera posició de l'*array*: `$($quadres[2])`.

En el segon s'ha fet correctament, cridant el mètode `get` passant com a argument la posició 0 de l'*array*, a continuació s'ha passat el node retornat a la funció jQuery i s'ha cridat el mètode `css` per canviar la propietat `background-color` a verd.

El mètode `get` retorna el node seleccionat de la posició; no es tracta d'un objecte jQuery. Un dels tipus de paràmetres que es pot passar a la funció jQuery és precisament un node. Així doncs, el que s'ha fet és crear un nou objecte jQuery a partir d'aquest node i seguidament cridar el mètode `css`.

A banda d'afegir o modificar propietats de l'estil CSS, també podeu afegir o eliminar classes d'aquests nodes. Per exemple:

```

1 <style>
2     div {

```

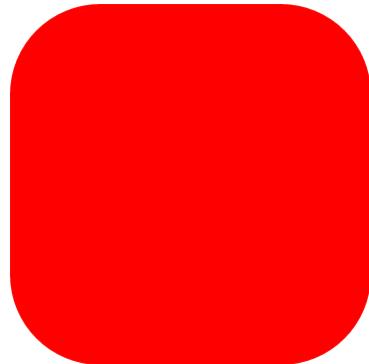
Si no es passa cap argument al mètode `get` retorna un *array* amb tots els nodes seleccionats.

```

3   width: 100px;
4   height: 100px;
5   margin: 10px;
6   background-color: lightgrey;
7 }
8
9 .vores {
10    border-radius: 50px;
11 }
12
13 .vermell {
14    background-color: red;
15 }
16
17 .gran {
18    width: 200px;
19    height: 200px;
20 }
21</style>
22<script>
23 $(document).ready(function() {
24     var $quadre = $('div');
25     $quadre.addClass('vermell');
26     $quadre.addClass('gran');
27 });
28</script>
29
30<div></div>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/WwvXMr, i el resultat a la figura 2.13.

FIGURA 2.13. Múltiples classes afegides a un element



En aquest cas s'han afegit dues classes CSS amb el mètode `addClass`, una anomenada `vermell` i una `gran`, que són afegides en temps d'execució i fan que el quadre tingui 200x200 px de mida i color vermell en lloc de gris.

Per eliminar-les és igual de fàcil, només s'ha de fer servir el mètode `removeClass`. Afegiu aquesta línia dintre de la funció:

```
1 $quadre.removeClass('vores');
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/jqPaKM, i el resultat a la figura 2.14.

FIGURA 2.14. Classe eliminada d'un element



Com que s'ha eliminat la classe que tenia associada inicialment, ja no són visibles les vores arrodonides. En cas de voler eliminar totes les classes només heu de cridar `removeClass` sense passar cap argument. Afegiu aquesta línia dintre de la funció:

```
1 $quadre.removeClass();
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/BKNmPy.

Events de jQuery

A banda del mètode `on` poden afegir-se *handlers* (funcions que són cridades en disparar-se l'*event*) fent servir mètodes específics com `ready` o `click`. Podeu trobar una llista de totes aquestes dreceres en el següent enllaç: goo.gl/U8yxFW.

Ara el quadre es veu de color gris clar i de la mida original, 100x100 px.

Altra característica molt important de jQuery és la facilitat amb la qual podeu afegir la detecció d'*events* (esdeveniments), com per exemple: si s'ha clicat un element o s'ha passat el cursor per sobre. Substituïu el codi JavaScript de l'últim exemple per aquest:

```
1 $(document).ready(function() {
2     var $quadre = $('div');
3
4     $quadre.on('mouseover', function(e) {
5         $(this).addClass('vermell');
6     })
7
8     $quadre.on('mouseout', function(e) {
9         $(this).removeClass('vermell');
10    })
11
12});
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/KzpyGa.

Si passeu el cursor per sobre del cercle veureu que canvia a color vermell, això és perquè es dispara l'*event* `mouseover`; en canvi, en sortir el cursor de sobre es dispara `mouseout` i s'elimina la classe `vermell`.

Primer es crida el mètode `on` i se li passa com a paràmetres una cadena amb el nom de l'*event* (o *events* separats per espais) que es volen escoltar, per exemple `mouseover`, i a continuació la funció que es cridarà.

Aquesta funció rep com a paràmetre un objecte que conté tota la informació de l'*event* (en aquest cas s'ha anomenat e, però podeu posar el nom que vulgueu), encara que no es fa servir en aquest exemple. Com ja heu vist en parlar del mètode each, this fa referència al node on s'ha produït l'*event*, així que per treballar amb aquest com un objecte jQuery heu de cridar la funció jQuery i passar el node com a paràmetre: \$(this).

A continuació crideu el mètode addClass o removeClass per afegir o treure la classe, respectivament, com ja s'ha vist a l'exemple anterior.

Tots els *events* funcionen de manera similar; vegeu com funciona l'*event click* i el mètode toggleClass per alternar l'activació de la classe cada vegada que es clica a sobre, i substituïu el codi JavaScript de l'últim exemple per aquest:

```

1 $(document).ready(function() {
2     var $quadre = $('div');
3
4     $quadre.on('click', function(e) {
5         $(this).toggleClass('vermell');
6     })
7
8 });

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/RaPjvz.

Ara, en fer clic sobre l'element es crida la funció que al seu torn crida el mètode toggleClass sobre l'element i afegeix o treu la classe vermell segons si aquesta es troba present o no. Aquesta funcionalitat es pot combinar amb les animacions i transicions de CSS3; canvieu a l'exemple anterior el codi CSS pel següent:

```

1 div {
2     width: 100px;
3     height: 100px;
4     margin: 10px;
5     float: left;
6     background-color: lightgrey;
7     transition: 1s ease;
8 }
9
10 .vores {
11     border-radius: 50px;
12 }
13
14 .vermell {
15     border-radius: 0px;
16     background-color: red;
17     transform: rotateZ(180deg);
18 }

```

Podeu veure aquest exemple en següent enllaç: codepen.io/ioc-dawm09/pen/aNOxNR, i el resultat a la figura 2.15.

FIGURA 2.15. Efecte en clicar els elements



Al primer bloc s'afegeix la propietat `transition`, amb una durada d'un segon, i `ease` com a funció d'interpolació per suavitzar-la. Això fa que en canviar qualsevol propietat, el canvi es produueixi progressivament al llarg d'un segon.

A la classe `vermell` s'han canviat les propietats `border-radius` per eliminar les vores arrodonides i s'ha afegit una rotació de 180º sobre l'eix Z.

Ara, en clicar sobre el cercle es pot veure com canvia de color, rota i es converteix en un quadrat. Tot això sense haver de modificar el codi JavaScript, només modificant el CSS. Com es pot apreciar en aquest petit exemple, podeu tenir dos llocs web amb exactament el mateix codi i només canviant el CSS aconseguir efectes i experiències completament diferents.

Vegeu un altre exemple de com combinar CSS i jQuery afegint animacions de CSS3 combinades amb els *events* `mouseover` i `mouseout`:

```
1 <style>
2   div {
3     width: 100px;
4     height: 100px;
5     float: left;
6     border: 1px solid lightgrey;
7     margin: 10px;
8     opacity: 0.5;
9     transition: 0.5s ease-out;
10    }
11
12  .pols {
13    background-color: blue;
14    animation-name: pols;
15    animation-duration: 0.5s;
16    animation-iteration-count: infinite;
17    animation-direction: alternate;
18  }
19
20  @keyframes pols {
21    from {
22      transform: scale(1.0);
23      opacity: 0.5;
24    }
25    to {
26      transform: scale(1.2);
27      opacity: 1;
28    }
29  }
30 </style>
31 <script>
32 $(document).ready(function() {
33   var $quadres = $('div');
34
35   $quadres.on('mouseover', function(e) {
36     $(this).addClass('pols');
37   })
38
39   $quadres.on('mouseout', function(e) {
40     $(this).removeClass('pols');
41   })
42 });
43 </script>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/reVbXb, i el resultat a la figura 2.16.

FIGURA 2.16. Efecte en clicar als elements



En aquest cas s'ha tornat a fer servir el mètode `on` amb els *events* `mouseover` i `mouseout` per detectar quan entra i quan surt el cursor dels quadres, i els mètodes `addClass` i `removeClass` per afegir o eliminar la classe `polS`, que afegeix:

- Un canvi de color del fons a blau.
- Una animació que s'ha definit amb el nom de `polS`, amb una durada de 0.5 s, que es repetirà indefinidament i amb la direcció de l'animació alterna, és a dir, que quan acaba torna enrere.

Queda pendent veure una altra de les característiques més potents de jQuery: com afegir i eliminar nodes del document i manipular el DOM:

```

1 <style>
2   .missatge {
3     border-radius: 5px;
4     padding: 10px;
5     font-size: 1.2em;
6   }
7
8   .error {
9     background-color: #f2dede;
10    color: #d9534f;
11    border: 1px solid #d9534f;
12  }
13
14  .exit {
15    background-color: #dff0d8;
16    color: #5cb85c;
17    border: 1px solid #5cb85c;
18  }
19
20  .info {
21    background-color: #d9edf7;
22    color: #5bc0de;
23    border: 1px solid #5bc0de;
24  }
25 </style>
26
27 <script>
28 $(document).ready(function() {
29   var $avisos = $('#avisos'),
30     $entrada = $('#entrada'),
31     $validar = $('#validar'),
32     $netejar = $('#netejar');
33
34   $validar.on('click', function(e) {
35     esborrarAvisos();
36
37     if ($.isNumeric($entrada.val())) {
38       afegirAvis('exit', 'Has introduït un nombre!');
39     } else {
40       afegirAvis('error', "S'ha d'introduir un nombre");
41     }
42
43     afegirAvis('info', 'El text introduït ha estat: ' + $entrada.val());
44   });

```

```

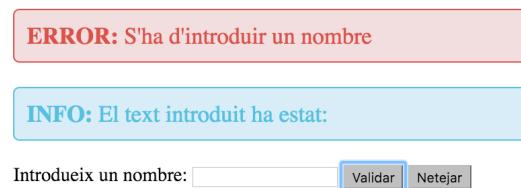
45
46     $netejar.on('click', function(e) {
47         esborrarAvisos();
48     });
49
50     function afegirAvis(tipus, missatge) {
51         var definicioNode = '<p class="missatge ' + tipus + '"><b>' + tipus.
52             toUpperCase() + ':</b> ' + missatge + '</p>';
53
54         $avisos.append(definicioNode);
55     }
56
57     function esborrarAvisos() {
58         $avisos.find('p').remove();
59     }
60 });
61 </script>
62
63 <div id="avisos"></div>
64 <label>Introdueix un nombre:
65 <input type="text" id="entrada"/>
66 </label>
67 <button id="validar">Validar</button>
68 <button id="netejar">Netejar</button>

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/eZVVmd.

Si proveu aquest exemple veureu que us mostrarà diferents missatges segons si la informació afegida ha passat la validació o no, com es pot apreciar a la figura 2.17.

FIGURA 2.17. Missatges de validació



Primerament, s'ha definit l'estructura del document, establint els id necessaris per poder identificar clarament on s'han de mostrar els avisos, quin és el text amb l'entrada de l'usuari, i quin és el botó per validar.

A continuació s'han afegit les classes CSS, una genèrica per a tots els avisos, anomenada `missatge`, que estableix les vores arrodonides, el padding i la mida de la font, i altres específics per canviar el color de l'avís segons si es tracta d'un tipus o un altre: `error`, `exit` i `info`.

El codi JavaScript no és gaire diferent del que s'ha vist fins ara, però inclou un parell de funcions pròpies:

JavaScript no proporciona cap mètode infal·lible per comprovar si una cadena es correspon a un nombre o no; per aquesta raó, és millor fer servir jQuery o implementar la nostra pròpria funció.

- `afegeixMissatge(tipus, missatge)`: afegeix el missatge del tipus passat com a argument dins del div amb `id="avisos"`.
- `esborrarAvisos()`: esborra tots els avisos.

Al principi es troba la declaració de les variables i l'assignació als tres elements que es volen controlar: el contenidor dels avisos, l'entrada de text i el botó.

A continuació es fa servir el mètode `on` per escoltar quan es dispara l'*event click* sobre el botó, i una vegada clicat es produeixen les següents accions:

- S'esborren tots els avisos `esborrarAvisos()`.
- Es comprova si el valor introduït és o no un nombre amb `$.isNumeric($entrada.val())`.
- Si és numèric, es crida `afegeixMissatge` indicant que el tipus serà `exit` i el missatge.
- Si no ho és, es crida també `afegeixMissatge`, però aquest cop el tipus serà `error`.
- En tot cas, es crida un cop més a `afegeixMissatge` per afegir un avís de tipus `info` indicant el valor introduït.

Com es pot apreciar, `$entrada` és l'objecte jQuery que conté el node corresponent al quadre de text. Per accedir al valor d'aquest (la propietat `value`) accedim cridant el mètode `val`, en aquest cas amb `$entrada.val()`.

Tota la feina referida a afegir els nodes es troba dins de la funció `afegeixMissatge`. En primer lloc, es crea la cadena de text que conté el codi HTML que es vol afegir, en aquest cas és:

- Un paràgraf (`<p>`).
- Amb les classes `missatge` i la que correspongui al tipus (`error`, `exit` o `info`).
- I el missatge passat com a argument, amb el tipus en majúscules (`tipus.toUpperCase()`) i negreta (``).

En concret, la cadena composta quedaria així per a un missatge d'error: `<p class='missatge error'>ERROR: S'ha d'introduir un nombre</p>`.

Una vegada creada la cadena, s'afegeix dins del node contingut en l'objecte jQuery `$avisos`, que correspon a l'element `div` amb `id="avisos"`. Per fer això només s'ha de cridar el mètode `append`, com es veu en l'exemple: `$avisos.append(definicioNode)`. En aquests exemples s'afegeixen sempre dos missatges; fixeu-vos que **no se sobreescriuen**, sinó que s'afegeix un darrere de l'altre.

La primera acció que es fa en clicar el botó *Validar* i el botó *Netejar* és esborrar els avisos; vegem que passa quan es crida la funció `esborrarAvisos`:

- Es crida el mètode `find` al qual es passa el paràmetre '`p`' per seleccionar tots els paràgrafs que es trobin dins de l'element contingut a l'objecte guardat a `$avisos`, que en aquest cas és l'element `div` amb `id="avisos"`.

Hi ha moltes maneres d'afegir nodes amb jQuery, i fer servir una cadena de text és només una d'aquestes.

Podeu trobar més exemples de com afegir elements amb jQuery en el següent enllaç: api.jquery.com/append.

Mètode "find"

El mètode `find` aplica un selector (com el de la funció `jQuery`), però en lloc de cercar a tot el document només cerca a partir de l'objecte a partir del qual es crida, per exemple: per cercar tots els paràgrafs dins d'un contenidor.

- Com que el retorn de `find` és un objecte jQuery, es pot continuar cridant mètodes de jQuery directament, així que es crida el mètode `remove`, que fa que s'eliminïn tots els nodes seleccionats, en aquest cas tots els paràgrafs.

Convé subratllar que quan **un objecte jQuery** conté més d'un node i es crida qualsevol dels seus mètodes, els canvis produïts (o cerques) s'aplicaran a tots ells, per exemple en cridar els mètodes `css` o `remove`.

Separar el comportament en funcions com `esborrarAvisos` i `afegirAvis` proporciona molts avantatges, entre d'altres:

- S'evita la duplicació de codi, la qual cosa fa més fàcil el manteniment, ja que si voleu fer canvis a com s'esborren els elements només heu de fer els canvis en un lloc de l'aplicació.
- Encapsula la implementació de com es porten a terme els canvis, de manera que es pot modificar sense haver de tocar el codi de l'aplicació (penseu que una aplicació complexa podeu tenir aquest codi repartit entre diferents fitxers).

Vegeu ara alguns exemples alternatius de com afegir i eliminar nodes, relacionats amb aquests últims canvis. Començant amb un exemple més estructurat de com crear nodes i modificar-los abans d'afegir-los al document; substituïu el codi de la funció `afegirAvis` pel següent:

```

1  function afegirAvis(tipus, missatge) {
2      var $ressaltat = $('</b>'),
3          $avis = $('

</p>');
4
5      $ressaltat.text(tipus.toUpperCase() + ': ');
6
7      $avis.addClass('missatge');
8      $avis.addClass(tipus);
9
10     $avis.append($ressaltat)
11     $avis.append(missatge);
12
13     $avisos.append($avis);
14 }


```

El mètode `append` accepta com a paràmetres cadenes de text, nodes i objectes jQuery.

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/bpVeZV.

El codi ara és molt més llarg. Fixeu-vos en el que s'ha fet, pas per pas:

1. S'han creat dos objectes jQuery, un que conté un node de tipus b (ressaltat) i un altre de tipus p, paràgraf. El codi que es passa com a paràmetres és HTML.
2. Per crear l'etiqueta ressaltada amb el tipus del missatge s'ha fet servir el mètode `text`, que afegeix com a text dins de l'element seleccionat el que s'ha passat per paràmetre; en aquest cas, el tipus concatenant-li el símbol ':' i un espai en blanc.

3. A l'objecte que conté l'avís s'hi ha afegit primerament la classe `missatge` i a continuació la classe corresponent al tipus d'avís amb el mètode `addClass`.
4. A continuació s'ha afegit l'objecte jQuery amb el ressaltat (`$ressaltat`) al node que contindrà tot l'avís (`$avis`) fent servir el mètode `append`.
5. Es crida un altre cop el mètode `append` per afegir a continuació el missatge, que és una cadena de text, i s'afegeix com a text pla.
6. Finalment, s'afegeix al contenidor d'avisos emmagatzemat a la variable `$avisos` l'objecte jQuery amb les classes, el ressaltat i el missatge afegits.

A primera vista pot semblar massa complicat fer-ho així, ja que amb el codi anterior només s'havia de passar una cadena de text, però serveix per adonar-se de la potència de jQuery per crear qualsevol estructura que necessiteu afegint nous elements, continguts i classes abans d'afegir-lo al document.

El següent exemple és molt més simple: es modificarà la funció `esborrarAvisos` per buidar el contingut del node contingut a l'objecte `$avisos` en lloc d'eliminar tots els seus fills un per un; substituïu el codi de la funció `esborrarAvisos` de l'exemple anterior per aquesta:

```
1 function esborrarAvisos() {
2   $avisos.empty();
3 }
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/dMYpaP.

Com es pot apreciar, en aquest exemple s'ha fet servir un mètode nou, el mètode `empty`. Aquest no accepta cap argument i s'encarrega d'eliminar tots els descendents dels elements continguts a l'objecte jQuery. És més simple que l'anterior i més efectiu, ja que aquest s'encarrega de fer l'eliminació completa tant d'aquests com d'altres elements, com poden ser la subscripció a *events*, dades, etc.

2.1.4 Elements interactius avançats: biblioteques i connectors

La biblioteca jQuery és molt popular, i per aquesta raó es poden trobar biblioteques i altres tipus de connectors basats en jQuery que el requereixen per al seu funcionament i que augmenten les seves funcionalitats, per exemple afegint nous mètodes als objectes jQuery retornats per la funció.

Una d'aquestes biblioteques és jQuery UI, que inclou nous elements per millorar les interfícies d'usuari.

Afegir un connector o una biblioteca és molt fàcil, s'han de carregar dos fitxers:

- El full d'estils del connector: aquest conté tots els estils necessaris; en connectors molt simples pot ser que no existeixi.

Documentació jQuery UI

Si voleu obtenir més informació sobre aquesta biblioteca i descobrir totes les seves opcions podeu visitar la pàgina oficial en el següent enllaç: jqueryui.com, on també podeu trobar el codi per descarregar, personalitzar la descàrrega i veure els tutorials que ofereixen per als diferents components.

- El fitxer amb el codi de la biblioteca o connector, igual que es fa amb la biblioteca jQuery o amb qualsevol altre fitxer amb codi JavaScript.

A l'hora de carregar recursos externs sempre s'han de carregar primer els fulls d'estil, a continuació les biblioteques JavaScript i finalment la resta de fitxers amb codi JavaScript, encara que és recomanable afegir tant les biblioteques com el codi propi al final del codi HTML abans del tancament de l'element body.

Vegeu primer com afegir la biblioteca jQuery UI al vostre codi fent servir una CDN (encara que això no és necessari a CodePen perquè es pot fer la carrega des de les opcions):

```

1  <head>
2    <link rel="stylesheet" href="//code.jquery.com/ui/1.11.2/themes/smoothness/
3      jquery-ui.css">
4    <script src="//code.jquery.com/jquery-3.1.1.js"></script>
5    <script src="//code.jquery.com/ui/1.11.2/jquery-ui.js"></script>
6    <script>
7      $(document).ready(function() {
8        $("input[type=submit], a, button")
9          .button()
10         .on('click', function(e) {
11           e.preventDefault();
12         });
13       });
14     </script>
15   </head>
16   <body>
17     <button>Un element 'button'</button>
18
19     <input type="submit" value="Un 'element' input de tipus 'submit'" />
20
21     <a href="#">Un enllaç</a>
22   </body>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/oxjBMM.

Si executeu el següent codi veureu que s'ha afegit dins de la capçalera de la pàgina:

- La càrrega del fitxer amb els fulls d'estil de la biblioteca fent servir el CDN de jQuery: `jquery-ui.css`.
- La càrrega de les biblioteques jQuery (sempre primer) i jQuery UI, també a partir del CDN.
- El codi JavaScript que comentarem a continuació.

És molt habitual representar els enllaços com botons, són molt més fàcils de modificar amb CSS i es poden afegir els *handlers* que necessiteu.

El contingut del bloc HTML és un botó, un element `input` de tipus `submit` i un enllaç, però la seva representació és idèntica per a tots tres i no s'ha fet servir cap etiqueta de classe ni codi CSS.

Encara que potser no sembla gran cosa, aquest és un canvi important. L'aspecte dels botons no està controlat completament per CSS, sinó pel navegador, i cada fabricant ha fet la seva pròpia implementació. Això provoca que l'aspecte de

botons per defecte no sigui homogeni entre navegadors i és molta feina fer els canvis pel vostre compte perquè s'han de tenir en compte molts aspectes.

Vegeu què fa el bloc de codi JavaScript: fins que el document no ha carregat completament no s'executa res, per això tota la implementació s'afegeix dins de la funció que és cridada pel mètode `ready` de jQuery, que una vegada disparada crea un objecte jQuery amb una selecció de botons, enllaços i els elements `input` de tipus `submit`: `$(“input [type=submit] , a, button”)`.

Com que jQuery presenta una **interfície fluida**, permet enllaçar crides a diferents mètodes una darrere l'altra; per veure-ho més clar s'acostuma a posar cada crida en una línia diferent.

En una **interfície fluida** els mètodes retornen el mateix objecte que fa la crida, de manera que poden encadenar-se crides a diferents mètodes una darrere l'altra. Podeu trobar més informació sobre les interfícies fluides en el següent enllaç: en.wikipedia.org/wiki/Fluent_interface.

Seguidament, es crida el mètode `button`, que és un dels mètodes afegits per la biblioteca jQuery UI als objectes jQuery. Això és el que fa que s'afegeixin totes les classes i funcionalitats adients als elements seleccionats per l'objecte jQuery.

A continuació s'afegeix el mètode `on` per escoltar l'*event click* com s'ha vist en exemples anteriors, però dins de la funció l'únic que es fa és cridar `e.preventDefault()`, sent `e` l'objecte que conté tota la informació de l'*event* i `preventDefault` el mètode que atura el comportament per defecte, de manera que en clicar a qualsevol dels tres elements no es fa cap acció.

Si inspeccioneu el codi HTML del primer botó amb les eines de desenvolupador (o qualsevol d'ells) veureu que s'han afegit una sèrie de classes i atributs que no hi són en el nostre codi HTML; per exemple, el contingut de l'element `button` modificat:

```
1 <button class="ui-button ui-widget ui-state-default ui-corner-all ui-button-
   text-only" role="button">
2   <span class="ui-button-text">Un element 'button'</span>
3 </button>
```

Cridar el mètode `preventDefault` és molt habitual en el cas dels botons i enllaços per poder afegir-los una funcionalitat diferent a la habitual.

En cridar el mètode `button`, aquest s'ha encarregat d'afegir totes les classes CSS (que pertanyen al full d'estils de jQuery UI) i elements necessaris per donar aquest aspecte.

Aquesta biblioteca ofereix una gran quantitat d'opcions, dividides en quatre grups:

- **Interaccions:** interaccions basades en el comportament del ratolí, com arrosseggar, deixar anar o canviar la mida d'un element.
- **Widgets:** controls que es reemplacen o afegeixen noves opcions a les interfícies d'usuari, com són els botons, les barres de progrés, les pestanyes, etc.

- **Efectes:** mètodes que modifiquen els aspectes i les posicions dels elements, incloent-hi els que afageixen i eliminan classes.
- **Utilitats:** usades per jQuery UI per a la creació d'interaccions i *widgets*.

Podeu trobar una llista amb milers de connectors per a la biblioteca jQuery en el següent enllaç: plugins.jquery.com.

En algunes ocasions necessitareu incorporar funcionalitats complexes molt determinades, per exemple per mostrar una galeria fotogràfica o afegir un reproductor de vídeo. En aquests casos, el que heu d'afegir és un **connector** (programa en JavaScript amb una funcionalitat molt concreta) especialitat per resoldre aquest tipus de problemes.

A diferència de les biblioteques, els connectors només inclouen una quantitat limitada d'opcions de configuració, de manera que són molt més simples d'utilitzar i molt més lleugers.

2.1.5 Desenvolupament de connectors per a jQuery

A continuació veureu com crear el vostre propi connector per a jQuery. En ocasions us trobareu que heu desenvolupat un component que utilitzeu molt sovint, i us pot interessar convertir-lo en un connector per facilitar la seva reutilització i manteniment.

Creareu un connector anomenat **arrodonir** que afegirà vores arrodonides a tots els elements als quals s'apliqui. Feu una primera prova per veure el comportament que es vol recrear:

```
1 <style>
2   div {
3     width: 100px;
4     height: 100px;
5     background-color: blue;
6   }
7 </style>
8 <script>
9   $(document).ready(function() {
10     $('div').css('border-radius', '50px');
11   });
12 </script>
13 <div></div>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/GZpOMP.

S'ha d'esperar que l'estructura del document estigui correctament carregada, se seleccionen tots els elements div i se'ls aplica l'estil border-radius amb un valor de 50 píxels.

Convertir aquesta funcionalitat en un connector és molt fàcil, només s'ha d'afegir una funció anomenada **arrodonir** a `$.fn`, i a partir d'aquest moment aquesta funció passarà a estar disponible a tots els objectes jQuery. Reemplaçeu el codi JavaScript pel següent:

```

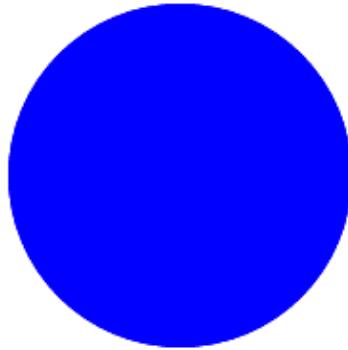
1 $.fn.arrodonir = function() {
2   this.css('border-radius', '50px');
3 }
4
5 $(document).ready(function() {
6   $('div').arrodonir();
7 });

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/mPeqXa.

Ara, quan es crida el mètode arrodonir als elements seleccionats per l'objecte jQuery, en aquest cas els elements div, s'aplicarà la propietat border-radius amb un valor de 50px, com es pot veure a la figura 2.18.

FIGURA 2.18. Element al qual se li ha aplicat el connector creat



Vegeu un exemple una mica més elaborat. Es calcularà l'alçada i l'amplada, i s'ajustarà el radi de les vores, de manera que si són iguals s'obtindrà un cercle, i si no, les bandes més estretes formaran un semicercle:

```

1 <style>
2   div {
3     width: 150px;
4     height: 120px;
5     background-color: blue;
6   }
7
8   div+div {
9     width: 250px;
10    height: 150px;
11    background-color: red;
12  }
13 </style>
14 <script>
15   $.fn.arrodonir = function() {
16     var height = this.height(),
17         width = this.width(),
18         radi = Math.min(height,width)/2;
19
20     this.css('border-radius', radi + 'px');
21   }
22
23 $(document).ready(function() {
24   $('div').arrodonir();
25 });
26 </script>

```

```

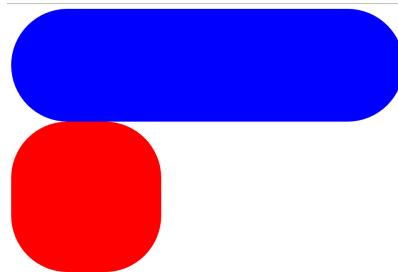
27
28 <div></div>
29 <div></div>
```

Si proveu a canviar la mida dels elements mitjançant el codi CSS a CodePen tingueu en compte que l'arrodoniment de les vores pot ser que no s'actualitzi correctament; això és una limitació de CodePen i no el comportament normal.

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/vGNWjV.

Fixeu-vos en aquest exemple que si la mida dels elements són diferents, els càlculs seran només correctes per al primer dels elements processats, encara que s'aplicarà la vora arrodonida a tots ells però amb el mateix valor, com es pot apreciar a la figura 2.19.

FIGURA 2.19. El mateix connector aplicat a múltiples elements



Vegeu un últim exemple: com ho faríeu si necessiteu poder passar arguments a aquest mètode? Molt fàcil, de la mateixa manera que a qualsevol altre mètode o funció. Substituïu el codi del bloc JavaScript pel següent:

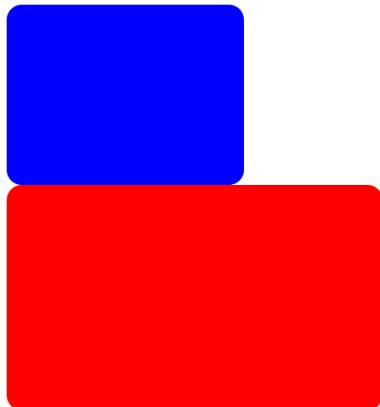
```

1 $.fn.arrodonir = function(radi) {
2     var height, width, radi;
3
4     if (!radi) {
5         height = this.height();
6         width = this.width();
7         radi = Math.min(height, width) / 2;
8     }
9
10    this.css('border-radius', radi + 'px');
11 }
12
13 $(document).ready(function() {
14     $('div').arrodonir(5);
15 })
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/ZWbame.

Simplement s'ha afegit un argument al mètode `arrodonir` anomenat `radi`, i dins del mètode es comprova si aquest **no està definit** (no s'ha passat cap valor en cridar el mètode) amb `if (!radi) {}`. Si no ho està, es fan els càlculs com a l'exemple anterior, i en cas que sí que estigui definit, es fa servir el valor per establir el `radi`. Si es passa el valor 5 (com a l'exemple) a la funció, el resultat seria el que es pot apreciar a la figura 2.20.

FIGURA 2.20. Elements modificats pel connector parametritzat



Només falta afegir-hi un petit detall. Per mantenir la interfície fluida de jQuery, el connector ha de retornar `this`: afegiu al final de la funció del connector `return this`. I per comprovar que realment es manté la interfície fluida, canviieu el codi de la funció cridada per `ready` per aquest:

```

1 $(document).ready(function() {
2   $('div')
3     .arrodonir(10)
4     .css('background-color', 'green');
5 });

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/ONyQmO.

Les vores dels elements `div` seran arrodonides 10 píxels, i el fons de tots dos elements serà de color verd.

Ja teniu el vostre connector preparat per fer-lo servir; podeu guardar-lo com un fitxer independent i carregar-lo com si es tractés d'una altra biblioteca, **sempre després d'haver carregat jQuery**, i el mètode `arrodonir` estarà disponible per a tots els objectes jQuery.

2.1.6 Introducció a JSON

El llenguatge de marques **JSON** (abreviatura de JavaScript Object Notation) permet guardar una col·lecció de dades de manera que és fàcilment interpretable pels humans. Vegeu aquest exemple, que es tracta d'un objecte de JavaScript creat a partir de la declaració literal:

```

1 var alumne = {
2   "nom": "Pere",
3   "cognoms: "Palau Torres",
4   "dni": "12345689X"
5 }

```

Fixeu-vos ara en com és aquest objecte en format JSON:

```

1  {
2    "nom": "Pere",
3    "cognoms: "Palau Torres",
4    "dni": "12345689X"
5  }

```

Origen de JSON

El llenguatge JSON va ser popularitzat i convertit en un estàndard per Douglas Crockford. Podeu trobar més informació sobre JSON i el seu promotor en el següent enllaç: en.wikipedia.org/wiki/JSON.

Són idèntics. Això és perquè el format d'intercanvi de dades JSON es basa en la declaració literal dels objectes JavaScript i, per tant, permet estructurar les dades de la mateixa manera. Per exemple, afegint *arrays*:

```

1  {
2    "nom": "Pere",
3    "cognoms: "Palau Torres",
4    "dni": "12345689X"
5    "signatures": [
6      "Programació bloc 1",
7      "Bases de dades bloc 1",
8      "Entorns de desenvolupament"
9    ]
10 }

```

O fins i tot altres objectes amb notació literal:

```

1  {
2    "periode": "2016/2",
3    "alumnes": [
4      {
5        "nom": "Pere",
6        "cognoms: "Palau Torres",
7        "dni": "12345689X"
8        "signatures": [
9          "Programació bloc 1",
10         "Bases de dades bloc 1",
11         "Entorns de desenvolupament"
12       ]
13     },
14     {
15       "nom": "Lluís",
16       "cognoms: "Martorell Ferrer",
17       "dni": "223353389D"
18       "signatures": [
19         "Programació bloc 2",
20         "Bases de dades bloc 2"
21       ]
22     }
23   ]

```

El format JSON només treballa amb les propietats de l'objecte; en cas que l'objecte contingui mètodes, aquests són descartats.

Si ho compareu amb altres llenguatges de marcat, com per exemple XML, es pot apreciar que un dels avantatges és que aquest és molt més simple i entenedor. Fixeu-vos com és la representació de l'exemple anterior amb format XML:

```

1 <institut>
2   <cursos>
3     <curs>
4       <periode>2016/2</periode>
5       <alumnes>
6         <alumne>
7           <nom>Pere</nom>

```

```

8      <cognoms>Palau Torres</cognoms>
9      <dni>12345689X</dni>
10     <signatures>
11       <signatura>
12         <nom>Programació bloc 1</nom>
13       </signatura>
14       <signatura>
15         <nom>Bases de dades bloc 1</nom>
16       </signatura>
17       <signatura>
18         <nom>Entorns de desenvolupament</nom>
19       </signatura>
20     </signatures>
21   </alumne>
22   <alumnes>
23     <nom>Lluís</nom>
24     ...
25   </alumne>
26 </alumnes>
27 </curs>
28 </cursos>
29 </institut>
```

Com podeu apreciar, en l'exemple anterior només s'ha afegit un alumne, però la quantitat de codi necessari ha estat molt més gran que per fer-ho en JSON i a més és més feixuc llegir la informació.

Un altre avantatge que presenta treballar amb JSON a JavaScript és que en tractar-se d'objectes nadius del llenguatge és molt fàcil afegir, modificar o eliminar dades, com es pot veure en el següent exemple:

```

1 // Es crea l'objecte alumne, que conté l'estructura de dades
2 var alumne = {
3   "nom": "Pere",
4   "cognoms": "Palau Torres",
5   "dni": "12345689X",
6   "signatures": [
7     "Programació bloc 1",
8     "Bases de dades bloc 1",
9     "Entorns de desenvolupament"
10   ]
11 };
12
13 // S'afegeix una nova propietat i li assignem un valor
14 alumne.edat = 24;
15
16 // Es modifica aquest valor
17 alumne.edat = 42;
18
19 // Finalment, s'elimina la propietat nom amb l'operador delete de JavaScript
20 delete alumne.nom
21
22 // Podeu trobar el contingut final a la consola
23 console.log(alumne);
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/EKQGxY, i el resultat a la figura 2.21.

FIGURA 2.21. Visualització a la consola de l'objecte alumne

```

Object {cognoms: "Palau Torres", dni: "12345689X", assignatures: Array[3], edat: 42}
  assignatures: Array[3]
    0: "Programació bloc 1"
    1: "Bases de dades bloc 1"
    2: "Entorns de desenvolupament"
    length: 3
  __proto__: Array[0]
  cognoms: "Palau Torres"
  dni: "12345689X"
  edat: 42
  __proto__: Object

```

Com podeu veure, el valor final per a la propietat `edat` és 42, i no existeix la propietat `nom` perquè l'hem eliminat.

Ara bé, encara que en aquest apartat no es treballa amb l'enviament ni la recepció de dades, és necessari saber que si voleu enviar aquest objecte a través de la xarxa, per exemple, fent servir un formulari, s'ha de convertir aquest en una cadena de text. I al contrari, pot ser que rebeu un objecte JSON com una cadena de text des del servidor i l'hàgiu de convertir en un objecte.

```

1 // Es crea l'objecte alumne, que conté l'estructura de dades
2 var alumne = {
3   "nom": "Pere",
4   "cognoms": "Palau Torres",
5   "dni": "12345689X",
6   "assignatures": [
7     "Programació bloc 1",
8     "Bases de dades bloc 1",
9     "Entorns de desenvolupament"
10  ]
11 };
12
13 // Conversió de l'objecte a cadena de text
14 var alumneAText = JSON.stringify(alumne);
15 console.log(alumneAText);
16
17 // Conversió la cadena de text a objecte
18 var textAAAlumne = JSON.parse(alumneAText);
19 console.log(textAAAlumne);

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/dMdwMO, i el resultat a la figura 2.22.

FIGURA 2.22. Visualització de l'objecte a la consola com a objecte i com a cadena de text

```

Object {"nom": "Pere", "cognoms": "Palau Torres", "dni": "12345689X", "assignatures": ["Programació bloc 1", "Bases de dades bloc 1", "Entorns de desenvolupament"]}
  assignatures: Array[3]
    0: "Programació bloc 1"
    1: "Bases de dades bloc 1"
    2: "Entorns de desenvolupament"
    length: 3
  __proto__: Object
  cognoms: "Palau Torres"
  dni: "12345689X"
  nom: "Pere"
  __proto__: Object

```

Com heu vist, JavaScript ens proporciona l'objecte global `JSON`, que exposa dos mètodes:

- **stringify:** converteix un objecte en una cadena de text en format JSON.
- **parse:** converteix una cadena de text en format JSON en un objecte JavaScript.

Amb aquests dos mètodes, i tenint clar com afegir, modificar i eliminar propietats d'un objecte, no heu de tenir cap problema per utilitzar-los com a estructures de dades en les vostres aplicacions.

2.1.7 Execució i verificació

Un dels majors problemes que us trobareu a l'hora de verificar el funcionament d'una aplicació web és la fragmentació del nombre de dispositius en el que s'ha de visualitzar.

Fins fa uns anys, la visualització de pàgines web estava lligada als equips d'escriptori, i per tant només calia preocupar-se de comprovar les pàgines i aplicacions web en un nombre limitat de navegadors. Tot i així, representava força dificultats, perquè sovint en un mateix equip no es podien instal·lar diferents versions d'un mateix navegador, o fins i tot els navegadors no eren compatibles amb diferents versions del mateix sistema operatiu.

Amb la proliferació dels telèfons intel·ligents, el problema ha empitjorat. No tan sols s'ha de comprovar en una combinació major de navegadors i sistemes operatius, sinó que és possible trobar casos en què fins i tot amb el mateix programari els dispositius poden tenir tipus de pantalles i sensors molt diferents, ja que tant pot tractar-se d'un dispositiu mòbil amb pantalla de 3" com d'una televisió intel·ligent amb una pantalla de 50".

Hi ha dues possibles solucions a aquest problema: la creació d'un **laboratori de dispositius** o **la simulació**. Cap de les dues és exhaustiva, i només permeten obtenir una certa seguretat que la nostra aplicació es visualitzarà correctament en determinades combinacions.

La creació d'un laboratori de dispositius no està a l'abast de tothom, ja que suposa una gran inversió en maquinari (s'han d'adquirir tots els dispositius per testejar) i, opcionalment, en programari. Existeixen alternatives gratuïtes i de pagament, tant per muntar la infraestructura per sincronitzar els dispositius com per al programari que els gestiona.

La simulació, en canvi, és més assequible, però té el desavantatge de no tractar directament amb dispositius reals. Quan parlem de simulació tenim opcions molt diferents:

- **Serveis remots de proves:** aquests serveis generalment són de pagament i ens permeten testejar la nostra web en una quantitat molt gran de dispositius. Per exemple, www.browserstack.com permet fer proves en més de 700 navegadors d'escriptori, Android i iOS.

Podeu trobar més informació sobre com muntar un laboratori de dispositius en el següent enllaç: goo.gl/tt8hcG.

Podeu trobar més informació sobre la sincronització multidispositiu en el següent enllaç: goo.gl/zkHpW1.

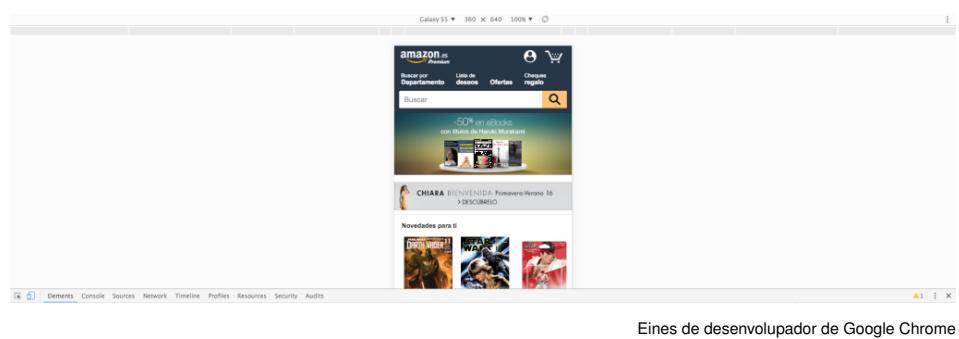
- **Virtualització:** aquest sistema suposa crear màquines virtuals al nostre equip amb diferents configuracions de sistema operatiu i navegadors. A banda de ser poc pràctic a l'hora de realitzar les proves, estem molt limitats quant a les possibilitats, ja que no tots els equips poden simular tots els sistemes operatius i navegadors. Per exemple, per emular el sistema operatiu macOS el vostre equip ha de tenir un processador Intel i3 o superior; per emular iOS cal fer servir macOS, i per emular Android de manera efectiva cal que el vostre equip faci servir processadors Intel, ja que són els únics que estan optimitzats per funcionar amb l'emulador proporcionat per Google (en el moment de redacció d'aquests materials).
- **Eines de desenvolupador:** les eines de desenvolupador dels diferents navegadors permeten simular la visualització en diferents mides; encara que no és ni de bon tros el mateix que simular tot un sistema operatiu ni els navegadors, sovint és suficient per cobrir la major part de les plataformes que us poden interessar.

Un dels objectius que s'han de plantejar en tots els projectes és que la pàgina o aplicació es visualitzi correctament quan es visiti a través de dispositius mòbils, fet que suposa comprovar-la com a mínim en els següents formats:

- dispositiu mòbil vertical
- tauleta vertical
- tauleta apaïsada
- escriptori

Afortunadament, aquestes quatre combinacions es poden comprovar fàcilment a través de les eines de desenvolupador, com es pot veure a la figura 2.23.

FIGURA 2.23. Visualització d'una pàgina simulant les dimensions d'un dispositiu mòbil



Si us fixeu en la figura 2.23 veureu que a la cantonada inferior dreta hi ha una icona que representa un mòbil i una tauleta; clicant sobre aquesta icona es passa al mode de disseny responsiu i podem seleccionar la visualització de la pàgina com si fossin diferents dispositius i resolucions.

A banda de **comprovar les resolucions**, heu d'assegurar-nos que la nostra aplicació funciona correctament com a mínim en els navegadors d'escriptori més utilitzats, que en el moment de redactar aquests materials són: Google Chrome, Mozilla Firefox, Safari, Opera i Internet Explorer.

En tractar amb elements multimèdia heu de tenir molt en compte que els dos principals problemes que us trobareu serà la incompatibilitat de formats de vídeo i àudio; per tant, sempre que feu servir els elements `audio` i `video` haureu de proporcionar com a mínim dos formats:

- **Àudio:** format MP3 i OGG.
- **Vídeo:** format MP4 i WebM.

Per pal·liar les limitacions que us trobareu en treballar amb navegadors antics disposeu de les següents opcions:

- **HTML:** entre els elements afegits a HTML5 hi ha alguns que només tenen funció semàntica, com per exemple `aside` o `article`, que es poden fer servir indistintament en navegadors antics (ja que es tracta de codi XML vàlid), i d'altres com `audio` i `video`, que inclouen la seva pròpia API. Aquests últims no poden funcionar en navegadors antics, i l'única solució actual és mostrar un missatge si no és possible utilitzar l'etiqueta:

```

1 <video>
2   <source src="fitxer_video.mp4" type="video/mp4">
3   <source src="fitxer_video.webm" type="video/webm">
4   Ho sento, el teu navegador no suporta vídeo d'HTML5
5 </video>
```

- **CSS:** en el cas dels fulls d'estil, la solució és afegir una propietat que funcioni universalment i a continuació afegir les funcions més avançades, de manera que s'aplicarà l'última acceptable. Per exemple, en el cas de voler afegir un degradat a un fons podeu fer servir la propietat `background-color` per afegir un color pla, i a continuació el codi de degradat propi dels diferents navegadors, acabant amb el de l'especificació de CSS3:

```

1 body {
2   background: #f0f9ff; /* Navegadors antics, color pla */
3   background: -moz-linear-gradient(-45deg, #f0f9ff 0%, #cbebff 47%, #a1dbff
4     100%); /* FF3.6-15 */
5   background: -webkit-linear-gradient(-45deg, #f0f9ff 0%, #cbebff 47%, #a1dbff
6     100%); /* Chrome10-25, Safari5.1-6 */
7   background: linear-gradient(135deg, #f0f9ff 0%, #cbebff 47%, #a1dbff 100%);
8     /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
9   filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#f0f9ff'
10     , endColorstr='#a1dbff', GradientType=1 ); /* IE6-9 fallback on
11       horizontal gradient
12 }
```

Podeu trobar més informació sobre *polyfill* en el següent enllaç:
goo.gl/h9NAhH.

- **JavaScript:** en el cas de JavaScript teniu dues opcions, la primera és implementar les funcions no disponibles en navegadors antics com a *polyfill*, de manera que si no són detectades les hi afegiu. S'ha de tenir en compte que aquesta solució sempre és més lenta que fer servir les funcions nadiues dels navegadors que sí les suportin. L'altra opció és fer servir les funcionalitats de biblioteques que ja implementen solucions per a navegadors antics, com jQuery amb les versions 1.x. Per exemple, per donar suport a Internet Explorer 6 i anteriors a l'hora d'enviar peticions AJAX al servidor s'havia de fer la següent comprovació:

```
1 if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
2     httpRequest = new XMLHttpRequest();
3 } else if (window.ActiveXObject) { // IE 6 i anteriors
4     httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
5 }
```

En canvi, si les peticions AJAX es fan a través de jQuery, no us heu de preocupar d'aquestes variacions, ja que ho gestiona la biblioteca.

Quan feu servir **biblioteques com Query** és recomanable emprar sempre les funcions i els mètodes que us proporcionen en lloc d'utilitzar JavaScript, perquè la biblioteca us assegura que el comportament estarà homogeneïtzat entre diferents navegadors i versions. Tenint en compte que jQuery és la biblioteca de JavaScript més utilitzada i que ofereix una gran quantitat de funcionalitats, és recomanable que l'estudieu pel vostre compte més enllà del que cobreixen aquests materials.

2.2 Casos pràctics d'integració de continguts multimèdia

Gràcies als nous elements afegits amb HTML5 (`audio` i `video`) és possible crear reproductors personalitzats fent servir només les tecnologies proporcionades pels navegadors, afegint llistes de reproducció que carreguin la informació de fonts externes com per exemple un servidor de vídeo extern o un fitxer de dades.

Un altre ús molt freqüent dels recursos multimèdia és la creació de bànders publicitaris, ja que aprofitant les característiques del llenguatge, en lloc de fer servir bànders estàtics amb codi HTML, es poden crear bànders dinàmics que mostrin un anunci o un altre segons les dades que s'hagin carregat (o incrustat).

En particular, els dos dels llenguatges de marcet més populars per transmetre aquesta informació són XML i JSON (JavaScript Object Notation). Un avantatge d'aquest últim és que es tracta del mateix format que fa servir JavaScript i per tant una estructura de dades en JavaScript és idèntica a la seva representació en JSON.

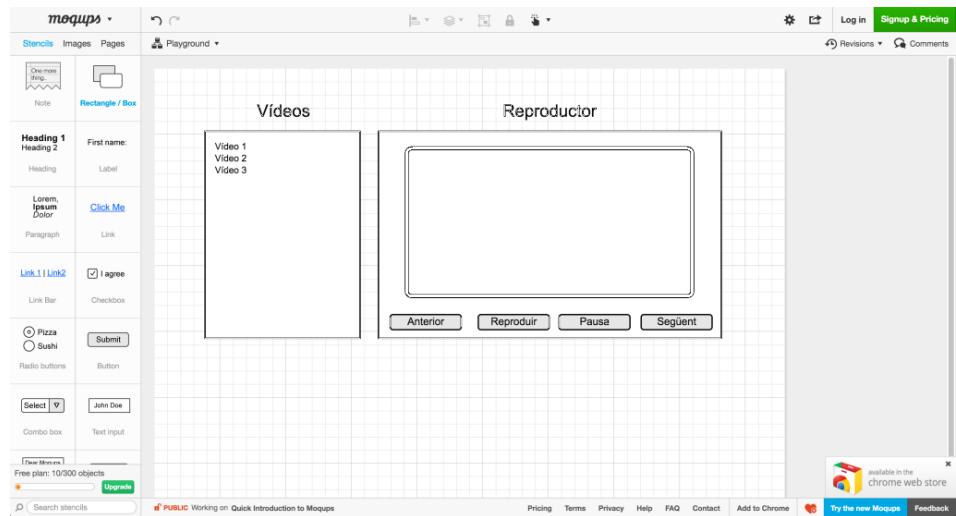
2.2.1 Creació d'un reproductor de vídeo

En primer lloc, es desenvoluparà un reproductor de vídeo, pas a pas, al qual s'integraran efectes de so i animacions amb CSS. Les característiques d'aquest reproductor seran les següents:

- A la secció esquerra mostrarà una llista generada dinàmicament a partir d'una estructura de dades JSON (en el nostre cas, un objecte literal de JavaScript)
- A la secció dreta mostrarà una caixa on es reproduirà el vídeo i a sota els botons de reproducció.
- En passar el cursor sobre qualsevol botó o element de la llista es reproduirà un so i s'executarà una petita animació.
- En fer clic sobre un element de la llista es reproduirà un so i començarà a reproduir-se el vídeo associat.
- En fer clic sobre un botó es reproduirà un so i es realitzarà una acció diferent, segons el botó clicat:
 - **Anterior**: es desplaçarà el vídeo seleccionat una posició cap enrere, de manera que si era seleccionat el primer vídeo, passarà a seleccionar-se l'últim, i començarà la reproducció.
 - **Següent**: al contrari que el botó *Anterior*, selecciona el següent vídeo de la llista, i si era l'últim, se selecciona el primer. A continuació comença la reproducció.
 - **Reproducir**: inicia la reproducció del vídeo seleccionat des del començament, i, si ja s'estava reproduint, tornarà a començar.
 - **Aturar**: si el vídeo estava reproduint-se l'atura, i, si estava aturat, continua reproduint des del mateix punt.

Prototip de la interfície del reproductor

El primer que s'ha de fer abans de començar a codificar la solució és fer un prototip de la interfície. D'aquesta manera, a l'hora de portar a terme la implementació tindreu clar com ha de funcionar l'aplicació. Aquest prototip el podeu dissenyar directament sobre paper o fer servir alguna eina de *wireframes* o *mockups* com la que es pot veure a la figura 2.24. Es recomana fer servir programari específic per a la creació d'aquests, ja que ofereixen elements neutres per crear les interfícies que no distreuen del seu objectiu.

FIGURA 2.24. Prototip dibuixat amb el programari 'moqups'

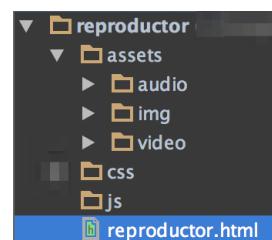
Es pot trobar el programari en línia 'moqups' en el següent enllaç (<https://moqups.com/>).

Una vegada tingueu clara la distribució de la interfície de la vostra aplicació podeu començar la següent fase de preparació.

Estructura de directoris i preparació de recursos

El següent pas és crear la estructura de directoris, preparar els recursos multimèdia necessaris i copiar-los als directoris pertinents.

És possible que l'estructura de directoris us vingui donada per les tecnologies que empreu; en aquest cas, fareu servir una estructura pròpia, com es pot veure a la figura 2.25.

FIGURA 2.25. Estructura de fitxers i directoris del reproductor

- El fitxer HTML es trobarà a l'arrel del projecte.
- Els fitxers CSS es trobaran dins del directori `/css`.
- Els fitxers amb codi JavaScript aniran dins del directori `/js`.
- Els fitxers de recursos multimèdia es trobaran dins del directori `/assets`, i dins d'aquests:
 - Els fitxers de vídeo dins de la carpeta `/video`.
 - Els fitxers d'àudio dins de la carpeta `/audio`.

- Els fitxers d'imatge dins de la carpeta */img* (encara que en aquest projecte no es farà servir cap).

Pas 1: preparació de l'estructura de la pàgina HTML

Arribats a aquest punt, ja podeu començar a codificar. Primer haureu de crear el fitxer HTML sense aplicar cap estil ni cap identificador, això ho fareu en el següent pas. Només heu de fixar-vos en el disseny del prototip i pensar com ha d'estar estructurat el codi.

Temporalment, es farà servir una llista (amb els elements *ul* i *li*) per representar els vídeos que poden seleccionar-se, ja que no s'afegirà la creació dinàmica fins més endavant.

Aquesta seria una possible implementació:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Reproductor</title>
5  </head>
6  <body>
7  <head>
8      <title>Reproductor</title>
9  </head>
10 <body>
11 <main>
12     <div>
13         <h1>Vídeos</h1>
14         <ul>
15             <li>Vídeo 1</li>
16             <li>Vídeo 2</li>
17             <li>Vídeo 3</li>
18         </ul>
19     </div>
20     <div>
21         <h1>Reproductor</h1>
22         <div>
23             <video width="430px" height="315px" type="video/mp4"></video>
24         </div>
25         <div>
26             <div>ANTERIOR</div>
27             <div>REPRODUIR</div>
28             <div>ATURAR</div>
29             <div>SEGÜENT</div>
30         </div>
31     </div>
32 </main>
33 </body>
34 </html>
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/GZmpWa.

I el resultat obtingut seria similar al que es pot veure a la figura 2.26.

FIGURA 2.26. Pas 1: estructura HTML del reproductor

Pas 2: afegir el full d'estil

Una vegada tingueu l'estructura llesta fareu servir el full d'estil per donar-li un format que s'ajusti al nostre prototip.

El primer que heu de fer és crear un fitxer de text pla anomenat reproductor.css, que guardareu dins del directori /css. A continuació, l'enllaçareu amb el document HTML afegint el següent codi dins de l'element head:

```
1  <link rel="stylesheet" href="css/reproductor.css" type="text/css" />
```

Recordeu que als **exemples de CodePen** no cal enllaçar els fitxers amb el codi CSS ni JavaScript, ja que aquest es troba a les diferents columnes. Per altra banda, tampoc no cal enllaçar la càrrega de la biblioteca jQuery, perquè està afegida a la configuració del Pen.

Seguidament, procediu a afegir els identificadors i classes a diferents seccions del codi HTML que us facilitarà l'assignació d'estils:

- identificador per a la llista
- identificador per al visor
- identificador per al reproductor
- identificador per als controls
- classes per als botons

Freqüentment, en treballar amb elements flotants, els elements que apareixen a continuació no es mostren correctament. Una de les solucions possibles és afegir un element div buit amb una classe que apliqui l'estil CSS clear:both. Per convenció, a aquesta classe se l'anomena `clear` o `clearfix`.

Ara només resta aplicar els estils apropiats:

```

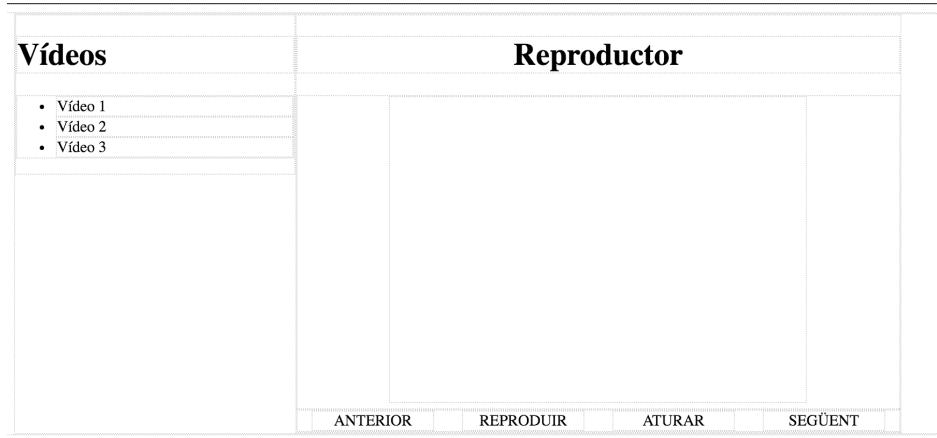
1  * {
2    border: 1px dotted lightgray;
3  }
4
5  main {
6    max-width: 960px;
7    margin: 0 auto;
8  }
9
10 #llista,
11 #visor {
12   float: left;
13 }
14
15 #llista {
16   width: 30%;
17 }
18
19 #visor {
20   width: 65%;
21   text-align: center;
22 }
23
24 #controls {
25   display: flex;
26   justify-content: space-around;
27 }
28
29 .boto {
30   flex-basis: 20%;
31 }
32
33 .clearfix {
34   clear: both;
35   border: none;
36 }

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/oxWjWP.

A la figura 2.27 podeu veure el resultat obtingut, molt més semblant al prototip.

FIGURA 2.27. Pas 2: reproductor després d'afegir el codi CSS



Fixeu-vos que s'ha afegit una vora a tots els elements del document:

```

1  * {
2    border: 1px dotted lightgray;
3  }

```

Aquesta vora s'ha afegit de manera temporal per ressaltar les delimitacions de tots els elements, i no formarà part del disseny final.

Pas 3: afegir fonts amb Google Fonts

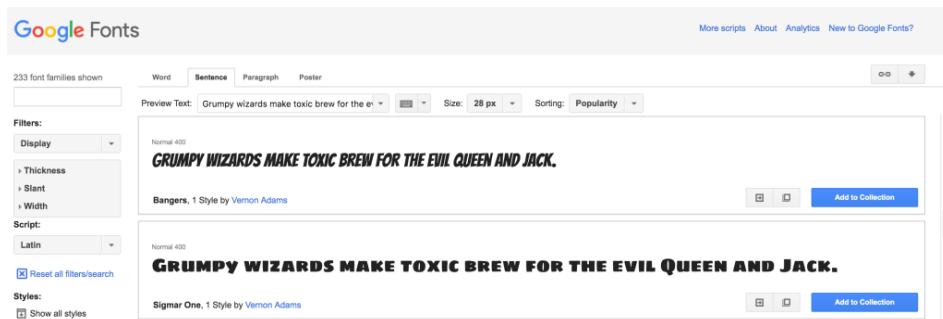
Les llicències de les fonts oferides per Google Fonts generalment són de tipus lliure o gratuïtes.

Una manera molt simple de fer els vostres títols i capçaleres més atractives és fer servir fonts externes, com les que ofereix Google Fonts.

Per afegir alguna d'aquestes fonts heu de visitar la pàgina de Google Fonts (vegeu la figura 2.28), i a continuació:

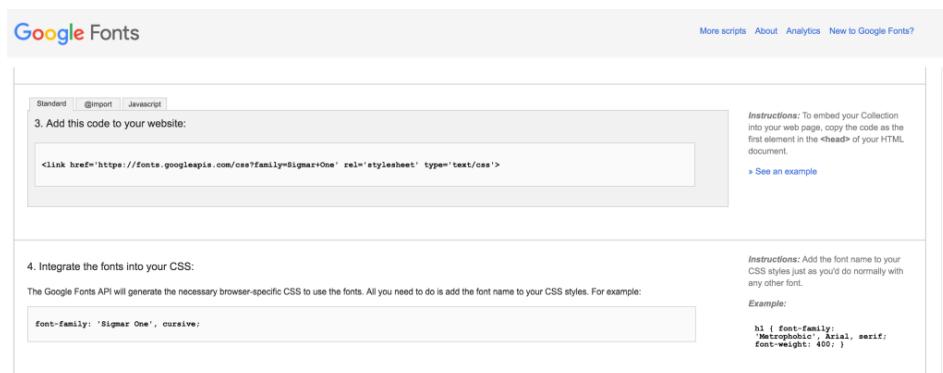
- Seleccioneu una de les fonts llistades, per exemple Sigmar One, de Vernon Adams.
- Afegiu-la a la col·lecció, fent clic al botó *Add to Collection*.

FIGURA 2.28. Selecció de la font de Google Fonts



- Feu clic al botó *use*, que us portarà a una altra pàgina on podreu veure més informació sobre la font, el seu pes i el codi per afegir tant al fitxer HTML i el codi CSS per fer-la servir, com es pot veure a la figura 2.29.

FIGURA 2.29. Visualització del codi HTML i CSS per utilitzar la font



Ara que ja teniu tota la informació que necessiteu, actualitzeu el vostre fitxer HTML amb el codi obtingut de la pàgina, i més a més afegiu el joc de caràcters per evitar problemes de representació:

```

1 <head>
2   <meta charset="utf-8">
3   <title>Reproductor</title>
4   <link href="css/reproductor.css" rel="stylesheet" type="text/css"/>
5   <link href="https://fonts.googleapis.com/css?family=Sigmar+One" rel="stylesheet" type="text/css">
6 </head>

```

Per assegurar que tots els caràcters de la pàgina es mostren correctament s'ha d'afegir a la capçalera l'etiqueta: `<meta charset='utf-8'>`.

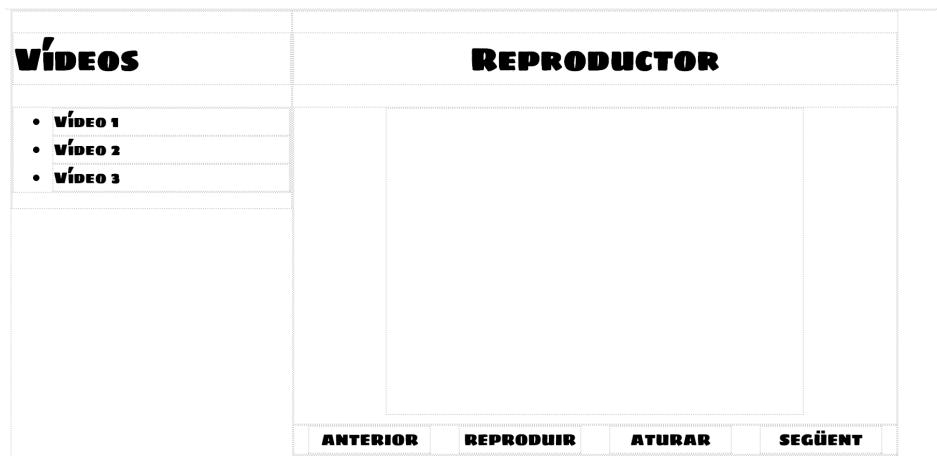
També s'ha de modificar el fitxer CSS per fer servir aquesta font a tots els elements, ja que no s'inclourà cap altre tipus d'element de text a banda de les capçaleres i els botons, i s'obtindrà el resultat que es pot veure a la figura 2.30.

```

1 * {
2   border: 1px dotted lightgray;
3   font-family: 'Sigmar One', cursive;
4 }

```

FIGURA 2.30. Pas 3: reproductor amb la font canviada



Pas 4: afegir icones amb Font Awesome

Per fer més clara la utilitat dels botons afegireu icones de Font Awesome. Com que precisament inclou un joc d'icones per a reproductors, facilita molt la feina.

L'avantatge de fer servir icones en lloc d'imatges és que les fonts funcionen com a imatges vectorials i, per tant, poden escalar-se a la mida que necessiteu sense perdre qualitat.

El primer que heu de fer per poder emprar les icones de Font Awesome és enllaçar amb el fitxer que conté el codi CSS:

```

1 <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome
      .min.css" rel="stylesheet">

```

Podeu trobar més informació sobre com utilitzar les icones de Font Awesome en el següent enllaç: goo.gl/54IXEX.

Tant **Google Fonts** com **Font Awesome**, a banda del fitxer CSS, realitzaran peticions extres per accedir als fitxers de fonts.

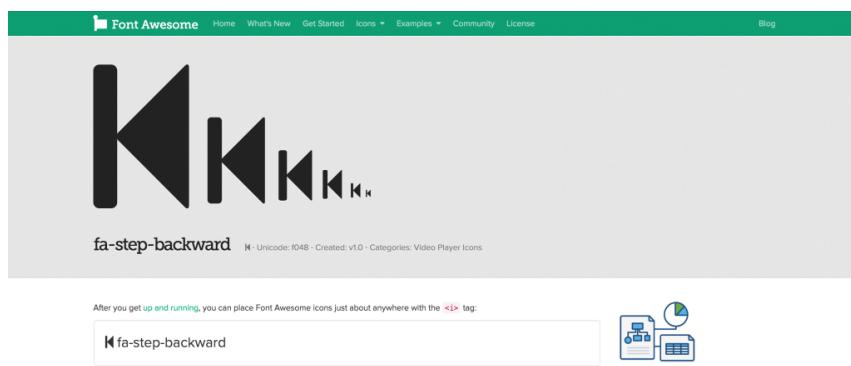
Una vegada enllaçat, ja podeu afegir les icones al reproductor; en aquest cas, s'ha decidit fer servir les icones:

- step-backward
- play
- pause
- step-forward

Per afegir un botó directament al codi HTML només hem de cercar la icona que us interessa dins del lloc web de Font Awesome i fer-hi clic. Us portarà a una altra pàgina, on trobareu el codi HTML que heu de fer servir.

La icona **step-backward** es pot trobar en el següent enllaç: goo.gl/MGm4PD. Allà es pot trobar el codi corresponent, `<i class="fa fa-step-backward"></i>` (com es pot veure a la figura 2.31), i la visualització en diferents mides.

FIGURA 2.31. Informació completa d'una icona de Font Awesome



Per augmentar la mida d'una icona només heu de fer servir una de les classes proporcionades per Font Awesome: `fa-2x` per duplicar la mida, `fa-3x` per triplicar-la, etc.

Reemplaçeu el codi dels controls pel següent, on s'han afegit les icones per al reproductor i un salt de línia per fer que el text quedi sempre en una nova línia:

```

1  <div id="controls">
2    <div class="boto"><i class="fa fa-step-backward fa-3x"></i><br>ANTERIOR</div>
3    <div class="boto"><i class="fa fa-play fa-3x "></i><br>REPRODUIR</div>
4    <div class="boto"><i class="fa fa-pause fa-3x"></i><br>ATURAR</div>
5    <div class="boto"><i class="fa fa-step-forward fa-3x"></i><br>SEGÜENT</div>
6  </div>

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/eZWpxV i la seva visualització a la figura 2.32.

FIGURA 2.32. Pas 4: icones de Font Awesome afegides al reproductor



Pas 5: canviar l'estil dels botons

El següent pas consisteix en donar estil als botons per fer-los més atractius. S'ha decidit fer servir dos tons de blau: un de clar per a l'estat normal i un de més fosc quan el cursor sigui a sobre del botó, i aplicar cantonades arrodonides. Al fitxer CSS canviem l'estil dels botons pel següent:

```

1 .botó {
2   flex-basis: 20%;
3   border-radius: 10px;
4   border: 2px solid #105F85;
5   padding: 5px;
6   color: white;
7   background-color: #2897E8;
8   cursor: pointer;
9 }
10
11 .botó:hover {
12   background-color: #105F85;
13 }
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/jqmWJv i la seva visualització a la figura 2.33.

FIGURA 2.33. Pas 5: Afegir estil als botons



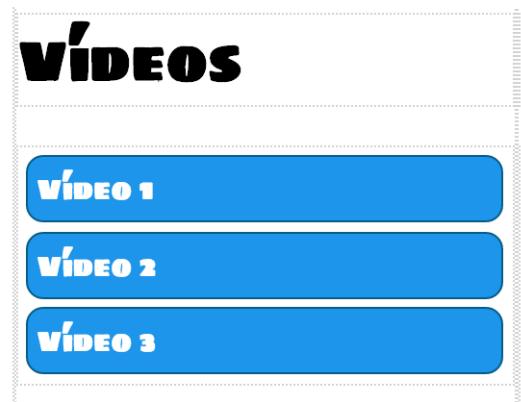
Pas 6: canviar l'estil de la llista

De manera semblant, modifiqueu l'estil de la llista aplicant el mateix tipus de vora, cantonades i colors:

```

1 #llista ul {
2   padding: 0;
3 }
4
5 #llista li {
6   list-style: none;
7   border-radius: 10px;
8   border: 2px solid #105F85;
9   padding: 5px;
10  color: white;
11  background-color: #2897E8;
12  cursor: pointer;
13  margin: 5px;
14 }
15
16 #llista li:hover {
17   background-color: #105F85;
18 }
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/eZWJwa i la seva visualització a la figura 2.34.

FIGURA 2.34. Pas 6: nou aspecte de la llista

Pas 7: refacció del full d'estil

La refacció consisteix a reestructurar un codi per millorar-lo sense modificar el seu comportament extern.

Us haureu adonat que el codi CSS tant del botó com dels elements de la llista és pràcticament idèntic. Arribats a aquest punt, és una bona idea fer una refacció per evitar repetir els blocs de codi. Una possible solució seria la següent:

```

1 #llista ul {
2     padding: 0;
3     margin: 0;
4 }
5
6 #llista li {
7     list-style: none;
8     margin: 5px;
9 }
10
11 .boto {
12     flex-basis: 20%;
13 }
14
15 .boto, #llista li {
16     border-radius: 10px;
17     border: 2px solid #105F85;
18     padding: 5px;
19     color: white;
20     background-color: #2897E8;
21     cursor: pointer;
22 }
23
24 .boto:hover, #llista li:hover {
25     background-color: #105F85;
26 }
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/BKRKar.

Pas 8: afegir efectes amb CSS3

Ara que ja teniu definits la llista i els botons, us resta afegir un fons per a tota la pàgina i fer algun petit retoc.

S'ha decidit fer servir un degradat mitjançant CSS3, i per ajudar-vos a generar el codi CSS podeu fer servir una de les múltiples eines en línia, per exemple la que es troba a www.colorzilla.com/gradient-editor. Només heu de seleccionar el tipus de

degradat, modificar-lo i copiar el codi que es genera com a propietat `background` de l'element `body` al fitxer CSS:

Cal tenir en compte que si només heu de preocupar-vos per navegadors moderns, el codi per generar el degradat es reduiria a `body {background: linear-gradient(top, rgba(167,207,223,1) 0%,rgba(35,83,138,1) 100%)}`.

```

1 body {
2     /* Permalink – use to edit and share this gradient: http://colorzilla.com/
   gradient-editor/#a7cfdf+0,23538a+100;Blue+3d+%238 */
3     background: rgb(167,207,223); /* Old browsers */
4     background: -moz-linear-gradient(top, rgba(167,207,223,1) 0%, rgba
   (35,83,138,1) 100%); /* FF3.6–15 */
5     background: -webkit-linear-gradient(top, rgba(167,207,223,1) 0%,rgba
   (35,83,138,1) 100%); /* Chrome10–25,Safari5.1–6 */
6     background: linear-gradient(to bottom, rgba(167,207,223,1) 0%,rgba
   (35,83,138,1) 100%); /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
7 }
```

Hi ha un inconvenient: en casos com aquest, en què la llargària de la pàgina no és suficient per cobrir-la sencera, el degradat s'atura on acaba el contingut. Una possible solució és fer servir un degradat lineal, i com a color del fons de l'element `html` el mateix color amb el qual acaba el degradat. D'aquesta manera, la transició és inapreciable:

```

1 html {
2     background: rgb(35,83,138);
3 }
4
5 body {
6     margin: 0;
7     padding-top: 20px;
8
9     /* Permalink – use to edit and share this gradient: http://colorzilla.com/
   gradient-editor/#a7cfdf+0,23538a+100;Blue+3d+%238 */
10    background: rgb(167,207,223); /* Old browsers */
11    background: -moz-linear-gradient(top, rgba(167,207,223,1) 0%, rgba
   (35,83,138,1) 100%); /* FF3.6–15 */
12    background: -webkit-linear-gradient(top, rgba(167,207,223,1) 0%,rgba
   (35,83,138,1) 100%); /* Chrome10–25,Safari5.1–6 */
13    background: linear-gradient(to bottom, rgba(167,207,223,1) 0%,rgba
   (35,83,138,1) 100%); /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
14 }
```

A continuació es canviarà el color de les capçaleres i l'alignació, i s'afegeixrà una ombra al text:

```

1 h1 {
2     text-align: center;
3     color: white;
4     text-shadow: 2px 2px 2px black;
5 }
```

Seguidament s'aplica com a fons un color de tipus RGBA (color de 24 bits amb transparència) i s'afegeix una ombra a tot el panell:

```

1 #llista, #visor {
2     background: rgba(0, 0, 0, 0.75);
```

Podeu trobar un generador d'ombres de text en el següent enllaç:
css3gen.com/text-shadow.

Podeu trobar un generador d'ombres de caixa en el següent enllaç:
www.cssmatic.com/box-shadow.

```

3 border-radius: 10px;
4 border: 1px solid black;
5 padding: 5px;
6 margin: 5px;
7
8 /* Ombra */
9 -webkit-box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);
10 -moz-box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);
11 box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);
12 }

```

En canvi, per al reproductor de vídeo s'afegeix un fons negre sólid, de manera que queda molt clar on es reproduiran els vídeos:

```

1 video {
2   background: black;
3   margin: 10px;
4 }

```

Finalment, elimineu la vora que es va afegir per comprovar fàcilment les delimitacions de cada element:

```

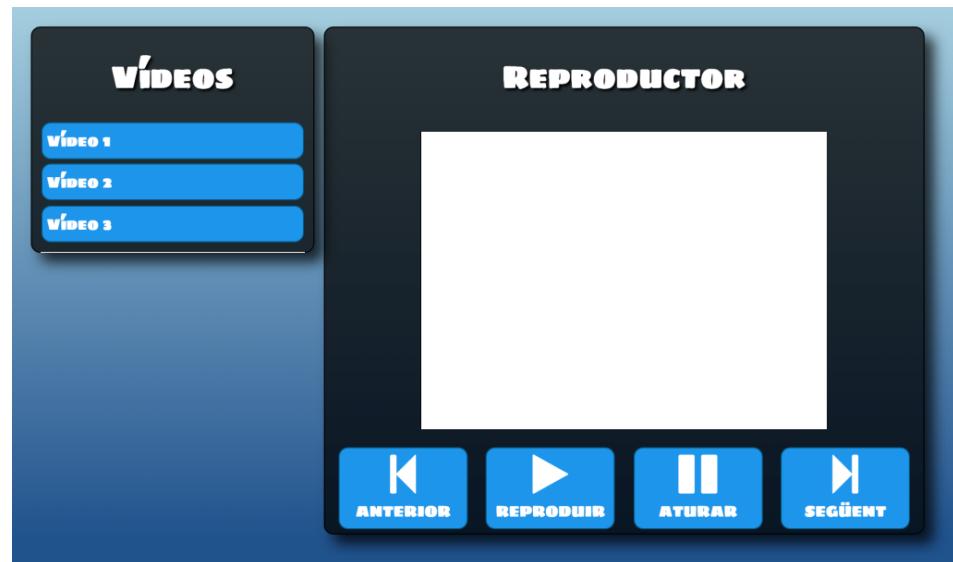
1 * {
2   font-family: 'Sigmar One', cursive;
3 }

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/aNWNvv.

Es pot veure el resultat obtingut després d'aplicar els canvis anteriors a la figura 2.35.

FIGURA 2.35. Pas 8: aspecte final del reproductor



Podeu trobar un generador de sons de 8 bits amb llicència CC0 (domini públic) en el següent enllaç: sfbgames.com/chiptone.

Pas 9: enllaçar la biblioteca jQuery i afegir efectes de so

Per continuar afegireu efectes de so als vostres botons: un quan el cursor estigui a sobre i un altre en fer-hi clic.

Primerament, heu d'obtenir els sons, per qualsevol mitjà; per exemple, sons descarregats de biblioteques de so amb llicència lliure, creats per vosaltres mateixos a través d'eines en línia o enregistrats i modificats amb Audacity.

Necessitareu obtenir dos fitxers d'àudio, que haureu de copiar dins del directori *assets/audio* del vostre projecte amb aquests noms:

- **selecciona.mp3**: aquest so es reproduirà en fer clic sobre un botó o element de la llista.
- **a-sobre.mp3**: aquest so es reproduirà quan passem el cursor per sobre d'un botó o element de la llista.

Una vegada tingueu els vostres fitxers d'àudio preparats, enllaceu la biblioteca jQuery i el fitxer amb el codi JavaScript, ja que fareu servir un fitxer extern per tenir el codi millor organitzat.

El primer que heu de fer és crear un nou fitxer de text buit dins de la carpeta *js* anomenat *reproductor.js*, i a continuació enllaceu tant la biblioteca jQuery com el vostre fitxer afegint el següent codi dins de la capçalera:

```
1 <script src="//code.jquery.com/jquery-3.1.1.min.js"></script>
2 <script src="js/reproductor.js"></script>
```

Sempre han d'enllaçar-se tots els **fitxers amb codi CSS** abans que els fitxers amb JavaScript, i quan treballem amb biblioteques, aquestes han de carregar-se sempre abans que els fitxers amb el codi que les fan servir.

Primerament s'afegirà la classe **sonor** als elements que vulgueu que produixin so en passar el cursor per sobre. Comenceu afegint-la als elements de la llista de vídeos:

```
1 <ul>
2   <li class="sonor">Vídeo 1</li>
3   <li class="sonor">Vídeo 2</li>
4   <li class="sonor">Vídeo 3</li>
5 </ul>
```

I a continuació, afegiu també la classe dels botons:

```
1 <div id="controls">
2   <div class="boto sonor"><i class="fa fa-step-backward fa-3x"></i><br>ANTERIOR
3     </div>
4   <div class="boto sonor"><i class="fa fa-play fa-3x "></i><br>REPRODUIR</div>
5   <div class="boto sonor"><i class="fa fa-pause fa-3x"></i><br>ATURAR</div>
6   <div class="boto sonor"><i class="fa fa-step-forward fa-3x"></i><br>SEGÜENT</div>
```

Seguidament, afegiu el següent codi JavaScript al fitxer *reproductor.js* que detectarà quan es dispara l'*event mouseenter* i reproduirà el so:

```
1 $(document).ready(function () {
2   // Codi JavaScript per detectar el mouseenter i reproduir el so
```

```

3     var a_sobra = new Audio('assets/audio/a-sobra.mp3');
4
5     $('.sonor').on('mouseenter', function() {
6         a_sobra.play();
7     });
8
9 });

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/xVdVpR.

Si proveu ara veureu que funciona, però no tal com s'espera. El so es reproduceix, però només torna a començar quan s'ha acabat la reproducció i no tan aviat com el cursor es col·loca sobre altre element.

Pas 10: creació d'un 'sound pool'

El so no sembla reproduir-se correctament. Aquesta és una limitació de l'element `audio` d'HTML. La solució és crear un conjunt de sons (*sound pool* en anglès), de manera que cada vegada que es demani reproduir un so s'agafarà el següent de la llista.

Utilització de 'pools'

En casos de jocs que poden tenir desenes de sons reproduint-se pràcticament al mateix temps, com IOC Invaders (github.com/XavierGaro/ioc-invaders), o d'imatges que es repeteixen, la reutilització d'aquests recursos representa una necessitat, ja que si no s'apliquen aquestes tècniques el nombre de quadres per segon es redueix dràsticament.

Haureu de substituir el codi del fitxer `reproductor.js` pels següents fragments. En primer lloc, afegireu la crida a la funció `inicialitzarSoundPool`, a la qual passareu dos arguments: el nom pel qual volem identificar el so i l'URL on es troba el fitxer d'àudio.

Un altre canvi és que en lloc de reproduir directament el so, ara es cridarà una altra funció, també pròpia, amb el nom `reproduirSo`, passant com a argument l'identificador que hem assignat al so.

```

1 $(document).ready(function() {
2
3     inicialitzarSoundPool('a_sobra', 'assets/audio/a-sobra.mp3')
4
5     $('.sonor').on('mouseenter', function() {
6         reproduirSo('a_sobra');
7     });
8
9 });

```

Reprodukció d'un nombre elevat de sons al mateix temps pot afectar negativament el rendiment de l'aplicació.

Seguidament s'afegeixen dues variables globals:

- `soundPool`: un objecte literal de JavaScript buit que es farà servir com a diccionari de dades.
- `MAX_SOUNDS`: una variable que es farà servir com si fos una constant, i que té la finalitat de limitar el nombre de sons idèntics que poden reproduir-se al mateix temps.

Ús de constants a JavaScript

Encara que a JavaScript 5 no existeix el concepte de constant, ens pot ser útil fer servir la convenció d'emprar algunes variables com si ho fossin i posar el seu nom en majúscules, com es fa en altres llenguatges.

```

1 var soundPool = {};
2   MAX_SOUNDS = 10;

```

Vegeu ara la implementació de la funció `inicialitzarSoundPool`, que té la finalitat de crear l'estructura de dades necessària per gestionar els sons i crear tots els elements d'àudio necessaris:

```

1 function inicialitzarSoundPool(nom, url) {
2   soundPool[nom] = {
3     sons: [],
4     actual: 0
5   };
6
7   for (var i = 0; i < MAX_SOUNDS; i++) {
8     soundPool[nom].sons.push(new Audio(url));
9   }
10 }

```

Com podeu veure, la funció és molt senzilla, però pot resultar una mica estranya si encara no es domina el llenguatge JavaScript.

El primer que s'ha fet és afegir a l'estructura de dades global `soundPool` un nou objecte literal. La clau per accedir a aquest objecte dins de `soundPool` serà el nom que s'ha passat com a argument, i l'objecte creat tindrà dues propietats:

- **sons**: un *array* que emmagatzemarà tots els elements d'àudio creats per reproduuir aquest so.
- **actual**: un *enter* que servirà per saber quin és l'element d'àudio que s'ha de reproduir en cada moment.

El mètode `push` és propi dels *arrays* de JavaScript i permet afegir un element al final de l'*array*.

A continuació s'utilitza un bucle `for` per crear tants elements d'àudio com indica la variable global `MAX_SOUNDS`, i s'afegeixen a l'*array* de l'objecte creat amb el mètode `push`: `soundPool[nom].sons.push(new Audio(url));`.

Fixeu-vos que tots els elements d'àudio així creats s'afegeixen a l'objecte guardat al *sound pool* amb el nom passat com a argument i amb el fitxer d'àudio especificat com a `url`.

Per exemple, si el nom és `a_sobra` i la `url` fos `assets/audio/a-sobra.mp3`, internament cada línia del bucle s'interpretaria així:

```

1 soundPool['a_sobra'].sons.push(new Audio('assets/audio/a-sobra.mp3'));

```

Per acabar, afegiu el mètode `reproduirSo`, que agafarà un nou element de so del *sound pool* cada vegada que es cridi fins a arribar a l'últim element, on tornarà a començar:

```

1 function reproduirSo(nom) {
2   var index = soundPool[nom].actual;
3   soundPool[nom].sons[index % MAX_SOUNDS].play();
4   soundPool[nom].actual++;
5 }

```

Podeu veure l'exemple complet en el següent enllaç: codepen.io/ioc-dawm09/pen/RaVQNV.

El primer que es fa és obtenir l'índex del so corresponent al nom passat com a argument que toca reproduir.

A continuació, s'aprofita la propietat del mòdul per evitar haver de fer la comprovació quan s'arriba al final de l'*array*. Atès que la mida d'aquest és igual a MAX_SOUNDS, es pot fer servir l'operació `index % MAX_SOUNDS` per obtenir sempre un valor dins del rang de l'*array* (entre 0 i MAX_SOUNDS - 1).

Finalment, s'incrementa el valor del so actual per deixar llist el *sound pool* per reproduir el següent.

Pas 11: reutilització de funcions

Ara que ja funciona correctament, afegiu el so per a l'*event click*, afegint al final de la funció creada per `ready` el següent codi:

```

1  inicialitzarSoundPool('selecciona', 'assets/audio/selecciona.mp3')
2
3  $('.sonor').on('click', function () {
4      reproduirSo('selecciona');
5  });

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-dawm09/pen/bpWLBj.

Com podeu veure, una vegada implementat el sistema de *pools* d'àudio és molt fàcil afegir qualsevol quantitat de sons. Només s'ha hagut de canviar el nom del so, l'URL del fitxer que es vol reproduir i l'*event* al qual es vol associar aquest nou so.

Pas 12: afegir animacions CSS

Podeu trobar una explicació detallada de com crear animacions amb CSS a l'apartat "Elements multimèdia al web: imatges, vídeo i àudio" de la unitat "Creació i integració d'elements multimèdia al web".

Per acabar amb els efectes dels botons i de la llista de vídeos s'inclourà una petita animació fent servir CSS; afegiu el següent codi al final del fitxer `reproductor.css`:

```

1  .sonor {
2      transition: 0.5s ease-out;
3  }
4
5  .sonor:hover {
6      animation-name: zoom;
7      animation-duration: 0.5s;
8      animation-direction: alternate;
9  }
10
11 @keyframes zoom {
12     from {
13         transform: scale(1.0) rotate(5deg);
14     }
15
16     to {
17         transform: scale(1.1) rotate(-5deg);
18     }
19 }

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/YqVeaE.

Pas 13: creació d'elements de la llista i reproducció de vídeo

Fins ara, la llista de vídeos que es mostrava era fixada pel codi HTML, però això no és gaire pràctic, ja que aquesta llista ha de ser dinàmica per poder afegir, eliminar o passar al següent vídeo.

En un cas real, segurament aquesta llista es carregaria fent servir AJAX, però per simplificar afegireu la informació dels vídeos directament al vostre fitxer reproductor.js amb una estructura de dades de tipus JSON, i es generaran els elements de la llista via codi.

El primer que fareu és eliminar els elements de la llista de vídeos i afegir un identificador per facilitar la feina d'afegir-los. La llista al vostre codi HTML ha de quedar així:

```

1 <div id="llista">
2   <h1>Vídeos</h1>
3   <ul id="videos">
4     </ul>
5 </div>
```

A continuació afegiu a la declaració de variables general del fitxer reproductor.js la informació dels vídeos com un *array* d'objectes de JavaScript creats amb *notació literal*, juntament amb una variable anomenada *selecciona* que inicialment tindrà el valor 0 per establir com a seleccionat el primer vídeo:

```

1 var soundPool = {},
2   MAX_SOUNDS = 10,
3   videos = [
4     {
5       titol: 'El primer vídeo',
6       descripcio: 'Aquesta és la descripció del primer vídeo',
7       url: 'assets/video/video-1.mov'
8     },
9     {
10       titol: 'El segon vídeo',
11       descripcio: 'Aquesta és la descripció del segon vídeo',
12       url: 'assets/video/video-2.mov'
13     },
14     {
15       titol: 'El tercer vídeo',
16       descripcio: 'Aquesta és la descripció del tercer vídeo',
17       url: 'assets/video/video-3.mov'
18     }
19   ],
20   seleccionat = 0;
```

Objectes a partir de notació literal

A JavaScript és possible crear objectes de forma literal, això facilita la creació d'estructures que es poden utilitzar com a diccionaris de dades, permetent tractar els elements emmagatzemats com a parells de *clau*-*valor*.

Seguidament, afegireu una funció per inicialitzar la llista de vídeos, la finalitat de la qual serà la següent:

- Recórrer a totes les posicions de l'*array* de vídeos.
- Crear un element de tipus li per a cada element de l'*array*.

- Afegir la classe `sonor` per conservar els efectes CSS afegits anteriorment.
- Afegir un identificador únic que estarà format pel mot `video-` i l'índex que li correspongui a l'element de l'`array`.
- Afegir el text a mostrar, que es correspondrà amb el valor del `titol`.
- Afegir l'atribut `title` amb el valor de `descripcio`; això fa que en posar el cursor sobre l'element es mostri la descripció del vídeo.
- Afegir el nou element a la llista de vídeos.
- Afegir la detecció de l'`event click` per establir aquest vídeo com el seleccionat i iniciar la seva reproducció.

Una vegada es té clar en què consisteix la implementació, fent servir jQuery trobareu que és pràcticament més simple que redactar-la, ja que per a cada una d'aquestes accions hi ha un mètode o funció de jQuery que facilita la tasca. Afegiu el següent codi al final del fitxer `reproductor.js`:

```

1  function inicialitzarLlistaVideos() {
2      $.each(videos, function (index, value) {
3          // Creem un nou node de tipus LI fent servir jQuery
4          var $node = $('- </li>');
5
6          // S'afegeix la classe sonor
7          $node.addClass('sonor');
8
9          // S'afegeix un identificador únic basat en el seu índex
10         $node.attr('id', 'video-' + index);
11
12         // S'afegeix el contingut de l'element
13         $node.text(value.titol);
14
15         // S'afegeix la descripció per mostrar quan deixem el cursor a sobre
16         // uns segons
17         $node.attr('title', value.descripcio);
18
19         // S'afegeix el node a la llista de vídeos
20         $('#videos').append($node);
21
22         // S'afegeix la detecció del esdeveniment click per establir aquest ví
23         // deo com el seleccionat i iniciar la reproducció
24         $node.on('click', function () {
25             seleccionat = index;
26             reproduirVideo();
27         });
28     });
29 }

```

Ara que ja teniu la funció que inicialitza la llista de vídeos només cal cridar-la. Haureu d'afegir la crida a aquesta funció just abans de la inicialització dels *pools*, perquè si ho afegiu després els elements de la llista no quedaran enllaçats amb aquests:

```

1  $(document).ready(function () {
2      inicialitzarLlistaVideos();

```

Finalment, només resta implementar la funció `reproduirVideo` i obtindreu un resultat com el de la figura 2.36:

```

1 function reproducirVideo() {
2     var video = videos[seleccionat];
3     $('video').attr('src', video.url);
4     $('video').get(0).play();
5 }

```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/ZWKPjy.

Aquesta funció és molt simple: primer s'agafen les dades del vídeo seleccionat de l'*array*, a continuació se selecciona l'element *video* i es canvia el seu atribuït *src* a l'URL del vídeo.

Fixeu-vos en un petit detall: el mètode *play* no pertany a jQuery, sinó que forma part de l'element *video* de la pàgina, així que primer s'ha de cridar el metode *get(0)*, que retorna el primer element de tipus *video* que es trobi a la pàgina, i a continuació cridar el seu mètode *play* per reproduir el vídeo.

FIGURA 2.36. Pas 13: versió final del reproductor amb vídeo



Pas 14: gestió del reproductor a través dels botons

Ja gairebé heu enllistit el vostre reproductor i només cal afegir la funcionalitat als botons. El primer que haureu de fer és afegir un identificador únic a cadascun d'aquests botons, modificant el codi HTML:

```

1 <div id="controles">
2     <div id="anterior" class="boto sonor"><i class="fa fa-step-backward fa-3x"></i><br>ANTERIOR</div>
3     <div id="reproducir" class="boto sonor"><i class="fa fa-play fa-3x "></i><br>REPRODUIR</div>
4     <div id="aturar" class="boto sonor"><i class="fa fa-pause fa-3x"></i><br>ATURAR</div>
5     <div id="seguent" class="boto sonor"><i class="fa fa-step-forward fa-3x"></i><br>SEGÜENT</div>
6 </div>

```

A continuació, heu d'afegir la detecció de l'esdeveniment *clic* per a cadascun d'ells; ho podeu fer a continuació de la inicialització dels efectes de so, associant cada botó amb una funció diferent:

```

1  $('#anterior').click(anteriorVideo);
2  $('#reproducir').click(reproducirVideo);
3  $('#ataruar').click(ataruarVideo);
4  $('#seguent').click(seguentVideo);

```

Es pot consultar la llista completa de dreceres per gestionar *events* en el següent enllaç: goo.gl/U8yxFW.

Aquest cop, en lloc de fer servir el mètode `on` de jQuery, s'ha fet servir el mètode `click`. En aquesta situació és indiferent, i es pot fer servir l'un o l'altre indistintament. jQuery facilita una extensa llista de dreceres per escoltar *events*: `click`, `dblclick`, `mousedown`, etc.

Si necessiteu escoltar **múltiples *events*** simultàniament heu de fer servir el mètode `on` de jQuery. Per exemple: `($('li').on('click dblclick', function() { alert("S'ha fet clic!"); });`.

Una altra diferència que es pot apreciar és que en lloc de cridar una funció com en els exemples anteriors s'ha posat només el nom de les funcions corresponents. En tots dos casos es tracta del que anomenem *callback*, funcions que són cridades per altres funcions. Per exemple, quan es dispara l'*event* associat com en aquest cas, quan es finalitza un temporitzador (amb les funcions `setInterval` o `setTimeout`) o quan retorna una petició AJAX del servidor amb èxit.

El següent pas és implementar aquestes funcions. Com que la funció `reproducirVideo` s'ha implementat anteriorment, només cal codificar les noves:

```

1  function aturarVideo() {
2      var video = $('video').get(0);
3
4      if (video.paused) {
5          video.play();
6      } else {
7          video.pause();
8      }
9  }
10
11 function anteriorVideo() {
12     seleccionat--;
13     if (seleccionat < 0) {
14         seleccionat = videos.length - 1;
15     }
16     reproducirVideo();
17 }
18
19 function seguintVideo() {
20     seleccionat++;
21     if (seleccionat === videos.length) {
22         seleccionat = 0;
23     }
24     reproducirVideo();
25 }

```

Recordeu que l'element `video` és un element afegit a HTML5 i compta amb una API pròpria que exposa mètodes com `play` i `pause`, i propietats com `paused`.

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/mPmgMV.

Vegeu amb detall aquestes funcions. En primer lloc, teniu la funció `ataruarVideo`. El seu comportament és diferent del de les altres, ja que ha de controlar l'estat de reproducció; per aquesta raó, el que s'ha fet és obtenir l'element de vídeo, i a partir

de la propietat `paused` determinar si cal aturar-lo (`video.pause()`) o reproduir-lo (`video.play()`).

Les funcions `anteriorVideo` i `seguentVideo` són pràcticament idèntiques, només redueixen o augmenten el valor de l'element seleccionat, canviant a l'última o primera posició respectivament si el valor es troba fora de rang, i a continuació inicien la reproducció del vídeo automàticament.

Pas: 15. afegir funcionalitat al teclat

Per acabar amb aquest exemple s'afegirà una funcionalitat més: quan es premi la tecla espai, el vídeo alternarà entre pausa i reproducció. Afegiu el següent codi a continuació del codi per gestionar els *events* dels botons:

```
1 $(document).keypress(function(evt) {  
2     if (evt.charCode==32) {  
3         aturarVideo();  
4     }  
5 });
```

Podeu veure aquest exemple en el següent enllaç: codepen.io/ioc-daw-m09/pen/qZmwed.

Com veieu, ha estat molt simple afegir aquesta funcionalitat. S'ha fet una subscripció per escoltar l'*event* `keypress` de tot el document, i quan es dispara es comprova si la propietat `charCode` té el valor 32 (que correspon a la barra d'espai). Si és així, es crida `atararVideo`, que l'aturarà o el reproduirà segons l'estat actual.

`keypress` és una drecera per
(on("keypress",
function(evt) {...}));

Cal tenir en compte que no tots els navegadors retornen la mateixa informació a l'objecte *event*, sobretot en referència als *events* de teclat i per tant és recomanable fer servir biblioteques per controlar aquest comportament.

A continuació podeu veure el llistat complet dels fitxers que componen el reproductor començant per `reproductor.html`:

```
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4     <meta charset="utf-8">  
5     <title>Reproductor</title>  
6     <link href="css/reproductor.css" rel="stylesheet" type="text/css"/>  
7     <link href="https://fonts.googleapis.com/css?family=Sigmar+One" rel="stylesheet" type="text/css">  
8     <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.5.0/css/font-awesome.min.css" rel="stylesheet">  
9  
10    <script src="//code.jquery.com/jquery-3.1.1.min.js"></script>  
11    <script src="js/reproductor.js"></script>  
12  
13 </head>  
14 <body>  
15 <main>  
16     <div id="llista">  
17         <h1>Vídeos</h1>  
18         <ul id="videos">  
19             </ul>  
20     </div>
```

```

21   <div id="visor">
22     <h1>Reproductor</h1>
23     <video width="430px" height="315px" type="video/mp4">
24     </video>
25     <div id="controls">
26       <div id="anterior" class="boto sonor"><i class="fa fa-step-backward fa-3x"></i><br>ANTERIOR</div>
27       <div id="reproducir" class="boto sonor"><i class="fa fa-play fa-3x "></i><br>REPRODUIR</div>
28       <div id="atarar" class="boto sonor"><i class="fa fa-pause fa-3x"></i><br>ATARAR</div>
29       <div id="seguent" class="boto sonor"><i class="fa fa-step-forward fa-3x"></i><br>SEGÜENT</div>
30     </div>
31   </div>
32   <div class="clearfix"></div>
33
34 </main>
35 </body>
36 </html>
```

Aquest és el contingut de reproductor.css:

```

1  * {
2    font-family: 'Sigmar One', cursive;
3  }
4
5
6 h1 {
7   text-align:center;
8   color: white;
9   text-shadow: 2px 2px 2px black;
10 }
11
12 html {
13   background: rgb(35,83,138);
14 }
15
16 body {
17   margin: 0;
18   padding-top: 20px;
19
20   /* Permalink - use to edit and share this gradient: http://colorzilla.com/
21      gradient-editor/#a7cfdf+0,23538a+100;Blue+3d+%238 */
22   background: rgb(167,207,223); /* Old browsers */
23   background: -moz-linear-gradient(top, rgba(167,207,223,1) 0%, rgba
24     (35,83,138,1) 100%); /* FF3.6-15 */
25   background: -webkit-linear-gradient(top, rgba(167,207,223,1) 0%,rgba
26     (35,83,138,1) 100%); /* Chrome10-25,Safari5.1-6 */
27   background: linear-gradient(to bottom, rgba(167,207,223,1) 0%,rgba
28     (35,83,138,1) 100%); /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
29 }
30
31
32 main {
33   max-width: 960px;
34   margin: 0 auto;
35 }
36
37 video {
38   background:black;
39   margin: 10px;
40 }
41
42 #llista, #visor {
43   float: left;
44 }
45
46 #llista {
```

```
42     width: 30%;  
43 }  
44  
45  
46 #visor {  
47     width: 65%;  
48     text-align: center;  
49 }  
50  
51 #controls {  
52     display: flex;  
53     justify-content: space-around;  
54 }  
55  
56 #llista ul {  
57     padding: 0;  
58     margin: 0;  
59 }  
60  
61 #llista li {  
62     list-style: none;  
63     margin: 5px;  
64 }  
65  
66 .boto {  
67     flex-basis: 20%;  
68 }  
69  
70 .boto, #llista li {  
71     border-radius: 10px;  
72     border: 2px solid #105F85;  
73     padding: 5px;  
74     color: white;  
75     background-color: #2897E8;  
76     cursor: pointer;  
77 }  
78  
79 .boto:hover, #llista li:hover {  
80     background-color: #105F85;  
81 }  
82  
83 .clearfix {  
84     clear: both;  
85     border: none;  
86 }  
87  
88 #llista, #visor {  
89     background: rgba(0, 0, 0, 0.75);  
90     border-radius: 10px;  
91     border: 1px solid black;  
92     padding: 5px;  
93     margin: 5px;  
94  
95     /* Ombra */  
96     -webkit-box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);  
97     -moz-box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);  
98     box-shadow: 10px 10px 20px 0px rgba(0,0,0,0.75);  
99 }  
100  
101 .sonor {  
102     transition: 0.5s ease-out;  
103 }  
104  
105 .sonor:hover {  
106     animation-name: zoom;  
107     animation-duration: 0.5s;  
108     animation-direction: alternate;  
109 }  
110  
111 @keyframes zoom {
```

```

112     from {
113         transform: scale(1.0) rotate(5deg);
114     }
115
116     to {
117         transform: scale(1.1) rotate(-5deg);
118     }
119 }
```

I finalment, el contingut de reproductor.js:

```

1 $(document).ready(function () {
2
3     inicialitzarLlistaVideos();
4
5     inicialitzarSoundPool('a_sobra', 'assets/audio/a_sobra.mp3')
6
7     $('.sonor').on('mouseenter', function () {
8         reproduirSo('a_sobra');
9     });
10
11
12     inicialitzarSoundPool('selecciona', 'assets/audio/selecciona.mp3')
13
14     $('.sonor').on('click', function () {
15         reproduirSo('selecciona');
16     });
17
18     $('#anterior').click(anteriorVideo);
19     $('#reproducir').click(reproducirVideo);
20     $('#aturar').click(aturarVideo);
21     $('#seguent').click(seguintVideo);
22
23     $(document).keypress(function(evt) {
24         if (evt.charCode==32) {
25             aturarVideo();
26         }
27     });
28 });
29
30 var soundPool = {},
31     MAX_SOUNDS = 10,
32     videos = [
33     {
34         titol: 'El primer vídeo',
35         descripcio: 'Aquesta és la descripció del primer vídeo',
36         url: 'assets/video/video-1.mov'
37     },
38     {
39         titol: 'El segon vídeo',
40         descripcio: 'Aquesta és la descripció del segon vídeo',
41         url: 'assets/video/video-2.mov'
42     },
43     {
44         titol: 'El tercer vídeo',
45         descripcio: 'Aquesta és la descripció del tercer vídeo',
46         url: 'assets/video/video-3.mov'
47     }
48 ],
49 seleccionat = 0;
50
51
52 function inicialitzarSoundPool(nom, url) {
53     soundPool[nom] = {
54         sons: [],
55         actual: 0
56     };
57
58     for (var i = 0; i < MAX_SOUNDS; i++) {
59         soundPool[nom].sons.push(new Audio(url));
```

```
60     }
61 }
62
63 function reproduirSo(nom) {
64     var index = soundPool[nom].actual;
65     soundPool[nom].sons[index % MAX_SOUNDS].play();
66     soundPool[nom].actual++;
67 }
68
69 function inicialitzarLlistaVideos() {
70     $.each(videos, function (index, value) {
71         // Creem un nou node de tipus LI fent servir jQuery
72         var $node = $('- </li>');
73
74         // Afegim la classe sonor
75         $node.addClass('sonor');
76
77         // Afegim un identificador únic basat en el seu índex
78         $node.attr('id', 'video-' + index);
79
80         // Afegim el contingut de l'element
81         $node.text(value.titol);
82
83         // Afegim la descripció per mostrar quan deixem el cursor a sobre uns
84             // segons
85         $node.attr('title', value.descripcio);
86
87         // Afegim el node a la llista de vídeos
88         $('#videos').append($node);
89
90         // Afegim la detecció de l'esdeveniment click per establir aquest vídeo
91             // com el seleccionat i iniciar la reproducció
92         $node.on('click', function () {
93             seleccionat = index;
94             reproduirVideo();
95         });
96     });
97 }
98
99 function reproduirVideo() {
100     var video = videos[seleccionat];
101     $('video').attr('src', video.url);
102     $('video').get(0).play();
103 }
104
105 function aturarVideo() {
106     var video = $('video').get(0);
107
108     if (video.paused) {
109         video.play();
110     } else {
111         video.pause();
112     }
113 }
114
115 function anteriorVideo() {
116     seleccionat--;
117     if (seleccionat < 0) {
118         seleccionat = videos.length - 1;
119     }
120     reproduirVideo();
121 }
122
123 function seguentVideo() {
124     seleccionat++;
125     if (seleccionat === videos.length) {
126         seleccionat = 0;
127     }
128     reproduirVideo();
129 }

```

Per finalitzar, podeu veure l'aspecte final del reproductor en funcionament a la figura 2.37.

FIGURA 2.37. Pas 15: resultat final



2.2.2 Creació d'un bàner amb carrega dinàmica de dades

Seguidament desenvolupareu un bàner que permetrà mostrar diferents anuncis de manera rotativa configurats a partir d'una estructura de dades JSON (més concretament un objecte literal de JavaScript). Les característiques que inclourà són:

- Creat a partir de l'element canvas existent al codi HTML.
- Configuració dels anuncis a partir de l'objecte JSON amb:
 - Imatge a mostrar.
 - Text que es mostrarà si es posa el cursor a sobre.
 - URL al qual es redirigirà en fer clic sobre l'anunci.
 - Canvi d'anunci periòdicament.

Aquest exemple és força interessant perquè soluciona un problema amb el qual us podeu trobar en el món laboral: les **còpies de pàgina a la memòria cau** (*cache*, en anglès). Per optimitzar el temps de càrrega de les pàgines és habitual fer servir algun sistema de memòria cau a la banda del servidor, i configurar les directives del servidor web per enviar els continguts amb temps d'expiració determinat. Això provoca que en visitar una pàgina múltiples vegades en un curt espai de temps, us podeu trobar que:

- Els continguts de la pàgina no s'actualitzen: la primera vegada que es demana la pàgina, el servidor fa totes les gestions necessàries per crear-la (per exemple, les consultes a la base de dades) i a partir d'aquest moment retorna sempre la mateixa pàgina. El sistema, per refreshar aquestes pàgines, depèn de l'aplicació del servidor i no hi ha res que pugui fer l'usuari en aquest sentit.
- Els fitxers que es descarreguen amb la pàgina, com poden ser imatges, fitxers amb codi JavaScript o CSS, fonts, etc., poden ser servits pel servidor amb una data d'expiració (per exemple, 30 dies a partir del moment que s'ha descarregat) i fins que no ha passat aquest temps o l'usuari esborra les dades del navegador no tornen a descarregar-se.

Si només necessiteu mostrar una sèrie d'anuncis fixos no hi hauria problema, però actualment el cas més habitual és haver de mostrar una gran quantitat d'anuncis diferents, i per tant incrustar els anuncis directament al codi HTML no és una solució vàlida.

Una possible solució és fer servir un servidor d'anuncis, això us permet aprofitar els avantatges de la memòria cau i mostrar diferents anuncis cada vegada, ja que la gestió de quins anuncis s'han de mostrar recau sobre el servidor que ens enviarà tota la informació necessària per mostrar els anuncis.

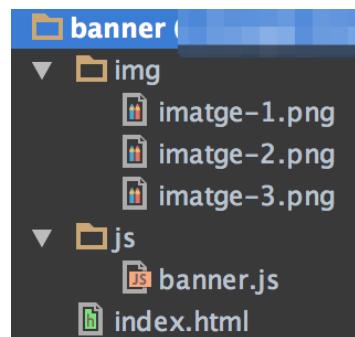
En el aques cas, el que e farà és afegir l'estructura de dades directament del fitxer amb el codi JavaScript, ja que com s'ha comentat anteriorment la implementació de peticions al servidor queda fora de l'abast d'aquests materials.

Una vegada es té clar el concepte hi ha moltes maneres de portar-lo a terme: es podria afegir directament codi HTML a la pàgina, o bé afegir un element `iframe` que contingúeu l'anunci. Aquesta implementació consistirà en utilitzar un element `canvas` per mostrar els anuncis.

En aquest exemple es farà servir una estructura simple que es pot veure a la figura 2.38. Els fitxers d'imatge aniran al directori `/img` i el fitxer JavaScript, a la carpeta `/js`.

Podeu trobar més informació sobre els serveurs d'anuncis o *Ad Servers* en el següent enllaç: en.wikipedia.org/wiki/Ad_serving.

FIGURA 2.38. Estructura de fitxers i directoris per al bàner



Afegiu aquest codi al fitxer `banner.html`; com podeu veure en aquest exemple, no es farà servir CSS, només s'enllaçarà amb la biblioteca jQuery i el fitxer de codi JavaScript:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Pàgina que conté el bàner</title>
6
7      <script src="//code.jquery.com/jquery-3.1.1.min.js"></script>
8      <script src="js/banner.js"></script>
9  </head>
10 <body>
11 <canvas id="banner" width="300" height="300">El teu navegador no suporta l'
    element canvas</canvas>
12 </body>
13 </html>
```

Vegeu ara el contingut del fitxer banner.js amb deteniment:

```

1 var TEMPS_ENTRE_ANUNCIS = 5 * 1000, // Temps en mil·lisegons
2   anuncis = [
3     {titol: 'Aquest és el primer anunci', imatge: 'img/imatge-1.png', url:
4      "http://www.example.com/anunci1"}, 
5     {titol: 'Aquí podeu veure el segon anunci', imatge: 'img/imatge-2.png',
6      url: "http://www.example.com/anunci2"}, 
7     {titol: 'I finalment, el tercer anunci', imatge: 'img/imatge-3.png',
8      url: "http://www.example.com/anunci3"}]
9   ,
10  anunciActual = 0;
```

El primer que s'ha afegit és una variable (emulant una constant), anomenada TEMPS_ENTRE_ANUNCIS, per facilitar la lectura del codi. Aquesta variable guardrà el temps que ha de passar entre en anunci i el següent en mil·lisegons.

A continuació podeu veure l'estructura de dades que s'ha fet servir per als anuncis d'exemple, un *array* d'objectes amb notació literal que contenen el títol, la imatge a mostrar i l'URL al qual redirigeixen en fer-hi clic.

Podeu trobar més informació sobre ES.next a es6-features.org.

I finalment, vegeu la **variable global** que indica quin és l'anunci actual.

Una vegada definida l'estructura de dades, s'afegeix la funció d'inicialització de l'aplicació. Es fa servir el mètode `ready` de jQuery per iniciar l'aplicació una vegada la càrrega del *DOM* s'hagi completat:

```

1 $(document).ready(function () {
2     var canvas = $("#banner").get(0), // Equivalent a: document.getElementById
3         ('banner');
4         context = canvas.getContext("2d");
5
6         rotarAnuncis(context);
7     });
```

En el bloc d'inicialització només ha calgut obtenir el context de l'element `canvas` que s'haurà de passar a la funció `rotarAnuncis` i que l'utilitzarà per iniciar la rotació d'anuncis.

Fixeu-vos que seria molt fàcil ampliar aquest codi per suportar múltiples àrees d'anuncis, només caldria canviar el comptador de l'anunci actual per suportar múltiples àrees i començar a rotar els anuncis dins de la funció d'inicialització amb les diferents àrees.

```

1 function rotarAnuncis(context) {
2     estableirAnunci(context);
3     setTimeout(function () {
4         anunciActual++;
5         rotarAnuncis(context);
6     }, TEMPS_ENTRE_ANUNCIS);
7 }
```

La funció `rotarAnuncis` també és molt simple: estableix l'anunci canvas del qual s'ha passat el seu `context` com a argument, i després inicia un temporitzador per incrementar el comptador d'anunci actual i cridar-se a si mateix.

Vegeu la implementació d'`estableirAnunci`, on es presenta un nou mètode jQuery, el mètode `unbind`:

```

1 function estableirAnunci(context) {
2
3     var anunci = anuncis[anunciActual % anuncis.length],
4         $banner = $('#banner').attr('title', anunci.titol);
5
6     // Ens assegurem que no existeixi ja un event de tipus click
7     $banner.unbind('click');
8
9     // Lliguem la funció a l'event click
10    $banner.click(function () {
11        window.open(anunci.url, '_blank');
12    });
13
14    dibuixarAnunci(anunci.imatge, context);
15 }
```

Fixeu-vos que s'ha fet ús de l'operació mòdul per evitar haver de controlar si el comptador d'anunci actual es troba fora de rang. A continuació podeu veure que es crida el mètode jQuery `unbind`, això serveix per eliminar qualsevol detector d'*events* lligat a l'*event* passat com a paràmetre (`click`). Si es fa així es poden provocar conflictes en fer clic a l'anunci, ja que es produeixen crides a les funcions anteriors.

Una vegada deslligades totes les funcions de l'*event* `click`, es procedeix a lligarlo a la funció que correspon, i que provoca l'apertura d'una nova pàgina, a una pestanya o finestra diferent, amb l'URL de l'anunci: `window.open(anunci.url, '_blank');`.

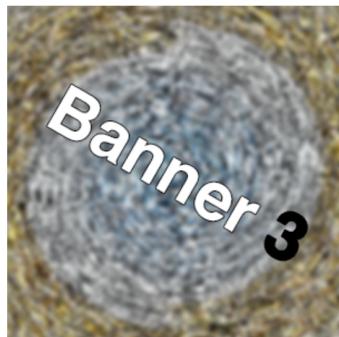
Finalment, només resta dibuixar l'anunci:

```

1 function dibuixarAnunci(imatge, context) {
2     var img = new Image();
3
4     img.onload = function () {
5         context.drawImage(img, 0, 0);
6     };
7
8     img.src = imatge;
9 }
```

Podeu veure l'exemple complet en el següent enllaç: codepen.io/ioc-daw-m09/pen/pywrJP i la seva visualització a la figura 2.39.

FIGURA 2.39. Bàner mostrant un dels anuncis



Aquest codi és molt simple, però conté un element molt important. Primer es crea un element de tipus `image` d'HTML5, i a continuació, **abans d'establir l'origen de la imatge**, s'afegeix una funció que serà cridada una vegada finalitzi la càrrega de la imatge, aquí és on afegim el codi que la dibuixarà al canvas. Si no s'afegeix el codi de la funció `onload` abans d'establir l'origen és possible que aquesta acabi de descarregar-se sense cridar la funció, especialment si aquesta es troba en local o emmagatzemada a la memòria cau del navegador.

S'ha de tenir en compte que els recursos de la pàgina es continuen carregant encara que hagi finalitzat la càrrega del DOM. Això vol dir que el codi pot començar a executar-se abans que les imatges s'hagin descarregat, la qual cosa provocarà que no es mostrin al canvas i per tant que no funcioni correctament. Per aquesta raó, **sempre que treballeu amb imatges i l'element canvas heu de gestionar l'esdeveniment `onload` de la imatge**, ja sigui creant un *gestor de descàrregues* o directament.

Finalment, s'estableix l'origen de la imatge, que es carregarà automàticament; a continuació dispararà l'*event* `onload` i es dibuixarà al canvas.

2.3 Els drets d'autor i l'ús de les llicències

Cal tenir en compte la qüestió dels drets d'autor i de l'ús de les llicències, ja que sense aquest coneixement podeu trobar-vos molts problemes a l'hora d'utilitzar elements que no hagin estat creats exclusivament per vosaltres.

Un punt molt important a l'hora de treballar amb elements multimèdia és que de vegades no es té clar si es pot fer servir un element en les nostres pàgines web o aplicacions, si podem editar-lo o si algú pot fer servir el vostre treball sense el vostre consentiment.

Fins i tot en el cas d'haver comprat una fotografia o cançó, això no us dóna automàticament el dret de fer-la servir, i per aquesta raó s'ha de tenir molt en compte sota quina llicència s'ha adquirit l'element i en cas de dubte consultar amb l'autor.

2.3.1 Drets de la propietat intel·lectual

Tota obra està subjecta als drets d'autor, de manera que en cas de dubte sempre es pot donar per suposat que una obra determinada està protegida i no es pot fer servir sense contactar primer amb el seu autor.

Per aquesta raó és imprescindible consultar sempre la llicència de qualsevol material que vulgueu fer servir, allà trobareu què se'n pot fer i què no. Un exemple són els llocs web de descàrregues de recursos que permeten als autors distribuir les seves creacions com a mostres del seu de treball, de demostració o de manera completament lliure.

Pel mateix motiu també és important incorporar la llicència d'ús als vostres treballs, ja que segons la llicència que feu servir se'n podrà fer un ús o un altre del nostre treball o les vostres publicacions.

Més informació sobre la propietat intel·lectual

Podeu trobar més informació sobre els drets de la propietat intel·lectual a la web del Ministeri d'Educació, Cultura i Esport: goo.gl/83dWgO.

2.3.2 Llicències

En el cas del disseny d'aplicacions web trobareu que a la vostra feina es poden aplicar diferents tipus de llicències; per una banda, part de la nostra feina pot ser creativa (dissensys d'imatges), i per una altra, normalment inclourà codi, i per tant s'aplicarien llicències de programari.

En tot cas, es poden distingir dos grans grups:

- **Llicències amb *copyright*:** si no s'especifica una altra cosa, tota obra és considerada automàticament amb *copyright*, no cal fer cap acció específica ni indicar-ho. Aquesta protecció permet a l'autor explotar la seva obra i cedir part dels seus drets a altres.
- **Llicències amb *copyleft*:** aquests tipus de llicències permeten a un autor renunciar als seus drets sobre l'obra, de manera que aquesta pot ser copiada, derivada, modificada i redistribuïda. Sempre s'ha d'incloure la llicència on s'especifica què pot i què no pot fer la persona que rep una còpia, per evitar per exemple que la redistribueixin sense aplicar-hi el *copyleft*. Dintre del *copyleft* podem trobar també llicències amb *copyleft* parcial. Una de les més conegudes d'aquest tipus és Creative Commons.

Drets d'autor i programari

S'ha de tenir en compte que en el cas del programari, pel que fa als drets d'autor, s'ha d'enregistrar com a "obra literària".

Un tipus de llicències molt utilitzades actualment són les **Creative Commons**. Aquestes comparteixen alguns punts amb el *copyleft*, ja que permeten als autors renunciar a part dels seus drets sobre les seves obres, de manera que poden ser compartides molt fàcilment.

Podeu trobar més informació sobre les llicències Creative Commons a cat.creativecommons.org.

Aquestes llicències tenen dos punts forts: per una banda, són molt fàcils d'entendre (com pot apreciar-se a la figura 2.40), i per una altra poden combinar-se per formar els següents sis tipus, més el de domini públic:

- **CC0:** domini públic. Renunciem a tot dret sobre l'obra (excepte als irrenunciables).
- **CC BY:** reconeixement. Permet distribuir, remesclar, retocar i crear a partir de l'obra, fins i tot amb fins comercials, sempre que s'acrediti l'autor.
- **CC BY-SA:** reconeixement-compartir igual. Com l'anterior, però l'obra s'ha de distribuir amb les mateixes condicions.
- **CC BY-ND:** reconeixement-sense obra derivada. Aquesta llicència no permet crear obres derivades, es pot distribuir l'obra comercialment però de manera íntegra i acreditant l'autor.
- **CC BY-NC:** reconeixement-no comercial. Amb aquesta llicència no és possible fer un ús comercial de l'obra.
- **CC BY-NC-SA:** reconeixement-no comercial-compartir igual. Aquest tipus no permet fer un ús comercial i la distribució s'ha de fer sota les mateixes condicions.
- **CC BY-NC-ND:** reconeixement-no comercial-sense obra derivada. Aquesta és l'opció més restrictiva, atès que no permet modificar l'obra ni comercialitzar-la.

FIGURA 2.40. Semàfor Creative Commons



Font: Marko Txopitea "Txopi" (commons.wikimedia.org)

Es poden trobar una gran quantitat de llicències de programari lliure, això és perquè no existeix un acord global sobre què és el programari lliure i el programari obert. Alguns desenvolupadors són partidaris que tot ha de ser lliure i gratuït, mentre que d'altres opinen que el codi ha de ser lliure però s'ha de pagar. És a dir, si una persona compra un programari determinat, aquest ha d'anar acompanyat del codi font.

Rendibilitzar el programari lliure

Una pràctica habitual per rendibilitzar els projectes de programari lliure és facilitar la base i cobrar després els serveis de consultoria, d'instal·lació, vendre llibres, desenvolupament de *plugins*, etc.

Un exemple de programari obert d'aquest tipus és **Wordpress**; per una banda, tenim el codi de forma gratuïta i, per una altra, es poden contractar els seus serveis d'allotjament i consultoria, comprar temes i *plugins* de tercers, llibres sobre diferents aspectes de Wordpress, etc.

Entre aquestes llicències, les més utilitzades són:

- **GNU General Public License (GPL) 2.0 i 3.0:** escrita originàriament per Richard Stallman, aquesta llicència garanteix a l'usuari final executar, estudiar, compartir i modificar el programari, raó per la qual s'ha d'adjuntar o oferir el codi font. Aplica el *copyleft*, de manera que les obres derivades només poden ser distribuïdes sota els mateixos termes de llicència, encara que permet vendre les obres derivades a qualsevol preu. Com que obligatòriament ha d'adjuntar el codi font, no permet incloure aquest codi junt amb programari privatiu.
- **MIT License:** creada pel Massachusetts Institute of Technology, aquesta llicència és molt semblant a la GNU GPL, però es pot distribuir com a programari privatiu i no adjuntar el codi font.
- **Apache License 2.0:** aquesta és com la del MIT, garanteix els mateixos drets que la GNU GPL amb la possibilitat de no adjuntar el codi font, però afegeix expressament els drets sobre patents per fer-lo servir.
- **BSD License 2.0:** molt similar a les dues anteriors, no és gaire recomanable usar-la, perquè pot ser confosa amb la versió BSD original, que és defectuosa. Per aquesta raó és més recomanable fer servir Apache License 2.0.

GNU GPL i programari privatius

Encara que aquesta llicència obliga a distribuir el programari amb el codi font, és possible llicenciar una mateixa obra o codi sota diferents llicències, de manera que seria possible distribuir el programari sota GPL i vendre'l a una companyia sota una llicència amb *copyright*.

En resum, el més habitual és que es faci servir una llicència GPL 3.0 si no voleu reservar-vos cap dret o una Apache License 2.0 si es vol reservar el dret de distribuir el codi font amb el programari lliure.

En cas de voler combinar diferents elements o codi (per exemple, llibreries), s'ha de tenir en compte que les seves llicències han de ser compatibles, aplicant-se sempre l'aspecte més restrictiu, però en cap cas es podran fer servir si una obliga a fer una cosa i l'altra la contrària. Per exemple, les quatre llicències de programari comentades anteriorment són compatibles amb GPL.

Es pot consultar una llista completa de llicències lliures al lloc web de GNU:
[www.gnu.org/licenses
/license-list.html](http://www.gnu.org/licenses/license-list.html)

Combinar dues imatges sota llicència Creative Commons

En el cas de tenir dues imatges sota llicència Creative Commons, si una és CC0 (domini públic) i l'altra és CC-BY (atribució), l'obra derivada podrà distribuir-se sempre que s'acrediti l'autor, perquè hi obliguen les restriccions CC-BY.

Combinar una llibreria amb llicència GPL i una altra de programari privatiu de tercers

En aquest cas hi ha una incompatibilitat, ja que la llicència GPL ens obliga a distribuir el codi font del programari i la llicència del *software* primitiu no ens permet distribuir-lo (i segurament tampoc ens l'han facilitat).

No seria possible combinar aquestes dues llibreries, i s'hauria de cercar una altra solució alternativa.

2.3.3 Recursos amb llicències fàcilment localitzables

A Internet podem trobar molts llocs de descàrregues, però només una minoria posen a la nostra disposició quines llicències s'apliquen a aquests recursos o si podeu reutilitzar-los pels vostres propis interessos.

Alguns d'aquests llocs estan dedicats a la venda d'aquests recursos, siguin dibuixos, música o fotografies, proporcionant una part de manera gratuïta per aconseguir més visites, i la resta de pagament (generalment lliures de regalies, *Royalty Free*).

Altres llocs on es poden trobar imatges amb llicències permissives són llocs dedicats a promocionar els treballs dels artistes.

A continuació teniu una llista de llocs on podeu trobar diferents tipus de recursos que o bé mostren les llicències d'aquests, en gran part permissives, o són de domini públic:

- **Fonts:**

- Google Fonts: www.google.com/fonts
- Dafont: www.dafont.com

- **Música i sons:**

- *as3sfxr* (generador de sons de 8 bits): www.superflashbros.net/as3sfxr
- Freesound (sons): www.freesound.org/browse
- Free Music Archive (música): www.freemusicarchive.org
- incompetech (música): incompetech.com

- **Imatges:**

- flaticon (imatges vectorials): www.flaticon.es
- Unsplash (fotografies en alta resolució, CC0): unsplash.com
- Behance: www.behance.net
- DeviantArt: www.deviantart.com

- **Codi:**

- CodePen (tots els *pens* són codi lliure): codepen.io
- GitHub (consultar llicències per a cada projecte): github.com