Departamento de Informática

PROYECTO FIN DE CARRERA

Accesibilidad a los contenidos audiovisuales en la Web a través de HTML5

Autor: Alberto Sánchez-Heredero Pérez

Tutor: Lourdes Moreno López

Leganés, Julio de 2011

Título: Accesibilidad a los contenidos audiovisuales en la Web a través de HTML5
Autor: Alberto Sánchez-Heredero Pérez
Director: Lourdes Moreno López

EL TRIBUNAL

Presidente:

_____

Vocal:

_____

Secretario:

_____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 1 de Julio de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mis padres por el apoyo que me han dado durante todos estos años de carrera, sin olvidarme por supuesto de mi hermano y mi novia que siempre han estado conmigo en los momentos difíciles.

A Lourdes por todo el ánimo que me ha dado durante la realización de este proyecto de fin de carrera, que a pesar de los retrasos e inconvenientes que hemos tenido, ha sido un placer haber trabajado con ella.

Resumen

Desde hace años asistimos a un continuo incremento de contenido audiovisual como vídeo en la Web, por ello que sea muy importante tener en cuenta requisitos de accesibilidad al incluirlo en la web para que todos los usuarios puedan acceder al contenido independientemente de sus características de acceso y contextos diversos de uso. Un contenido vídeo en la web debe ir acompañado de manera sincronizada de alternativas como subtitulado y audiodescripción entre otros, pero además el agente de usuario (reproductor) a través del cual los usuarios acceden debe ser también accesible. Como solución universal está el nuevo estándar HTML5 aun en desarrollo que proporciona algunas ventajas para la accesibilidad en lo que a contenido audiovisual se refiere. En este proyecto se presenta un estudio de estándares a cumplir para incluir contenido audiovisual en la Web, se valora si el nuevo estándar HTML5 da soporte y como caso de estudio se ha desarrollado un reproductor accesible en HTML5.

Palabras clave: Accesibilidad web, multimedia, vídeo, discapacidad, diversidad funcional, diseño web, desarrollo web, agente de usuario web, reproductor, estándares.

Abstract

In recent years, we have seen a continuous increase of audiovisual content like video on the Web. It is very important to consider accessibility requirements when the video is included in the web page enabling users to have access to the content, regardless of users access features and diverse contexts of use. Video content on the Web must be accompanied by synchronized alternatives such as caption and audio description, furthermore the user agent (player) must be accessible too. A universal solution is the new standard HTML5, although it is still under development. It provides some advantages for accessibility as far as audiovisual content is concerned. In this project, a study is presented, which includes standards and requirements that need to be fulfilled when there is audiovisual content on the Web. A case study with a web accessible player has been developed in order to assess if the new HTML5 standard really offers accessibility requirements support.

Keywords: Web accessibility, multimedia, video, disability, Web design, Web developed, User agent, player, standards.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción y objetivos

1.1. Introducción

En la última década, hemos sido testigos del gran cambio que ha provocado Internet en nuestras vidas. No teníamos ni idea de cómo un invento que se creó para unos pocos, ha sido una revolución para muchos y que ha hecho que Internet sea un elemento imprescindible para el día a día, ya sea en el ámbito personal como en el profesional.

En un principio, el origen de Internet fue la investigación de computadores más avanzados por parte del Departamento de Defensa de los EE.UU, pero con el paso del tiempo se fue popularizando y extendiendo a lo largo del mundo y se hizo público con el fin de darse a conocer y hacerse accesible para cualquier ciudadano.

Poco a poco se fueron investigando las características que ofrecía y fueron surgiendo lenguajes para dar soporte a esta nueva forma de comunicarse, y el principal fue el lenguaje de Marcado de Hipertexto o *HyperText Markup Language* (HTML). Éste nuevo lenguaje dio la posibilidad de compartir información con el resto del mundo, principalmente por texto, ya que éste lenguaje está basado en texto (o hipertexto como se denomina técnicamente), pero con el paso del tiempo surgieron nuevos lenguajes destinados a complementar al HTML, como son JavaScript, Flash, etc. que permitieron compartir diversos tipos de información, ya no sólo de texto sino imágenes, fotos, videos, animaciones, juegos, es decir, contenido multimedia que dotaba a los sitios web de un contenido más vistoso y agradable de compartir que el mero hecho de leer sin más.

Por tanto, todos estos nuevos lenguajes necesitaban unas normas que los programadores siguiesen para que la Web no sea un caos. Para garantizar la accesibilidad en la Web existe la *Web Accessibility Initiative* (WAI), iniciativa del *World Wide Web* (W3C) que ofrece unas pautas y reglas para permitir el acceso a contenido web a todo tipo de usuarios además de proponer normas a seguir para la estructura de una página web, y proporciona estándares de

distintos lenguajes como HTML, CSS (*Cascading Style Sheets*), etc., fundamentales para una buena construcción de sitios web. El Consorcio W3C es una comunidad internacional donde las organizaciones miembro, personal y el público en general, trabajan conjuntamente para desarrollar estándares web [W3C, 2011 c]. Desde entonces, la accesibilidad se convirtió en algo indispensable, y además de que una página web tenga un diseño de interfaz atractivo, su contenido debe ser accesible para cualquier tipo de persona.

El objetivo de este proyecto es investigar la accesibilidad en la Web en relación al contenido multimedia de tipo audiovisual en una nueva versión de lenguaje de marcado aun en desarrollo y en camino de estandarizarse como es el nuevo HTML5. Se describirá de manera extensa, y se indicarán las nuevas características que aporta a la accesibilidad en la Web, profundizando en las nuevas formas que propone para ofrecer recursos de accesibilidad en el contenido audiovisual.

1.2. Introducción y motivación

En este documento, memoria de proyecto de fin de carrera, se va a ofrecer documentación sobre la accesibilidad web en el ámbito de la multimedia, más concretamente en un elemento que es de sobra conocido por cualquier persona asidua a la Web, y que últimamente es más común de lo que fue hace unos cuantos años, y éste es el video.

El vídeo en la Web se ha convertido en una forma muy fácil y rápida de transmitir información de unas personas a otras, sobre todo gracias al auge de sitios web especializados en la difusión de vídeos, siendo el más conocido YouTube[1]. Uno de los objetivos de este tipo de sitios web, sería el de llevar su contenido al mayor número de personas posibles, independientemente de su habilidad y características, es decir, que sea lo más accesible posible.

Accesibilidad en el ámbito web, significa que cualquier persona que quiera acceder a un contenido en la Web, no tenga ninguna barrera o impedimento para acceder a dicho contenido, en este caso, contenido multimedia. Estas barreras pueden ser múltiples debido a las características de acceso del usuario, pero también pueden ser barreras de software como código mal implementado o no dirigido a la accesibilidad, como la ausencia de subtitulado en video, audiodescripción o factores técnicos como acceso por teclado o la obligación de contar con JavaScript en el navegador, etc.

La motivación que ha llevado a la realización de este proyecto es la búsqueda de la accesibilidad a través de nuevas tecnologías, más concretamente de HTML5. Lo que se busca es una nueva solución tecnológica para aportar subtítulos y audiodescripción dirigida a personas con discapacidad, ya que debido a la ingente cantidad de información que hay actualmente de contenido multimedia, concretamente en video, es necesario que esta información sea accesible a todo tipo de personas.

1.3. Objetivos

Los objetivos que se persiguen en este proyecto son los siguientes:

[1] http://www.youtube.com/

1) Llevar a cabo un estudio de la cuestión de los siguientes puntos principalmente:
   a) Introducción al HTML5
   b) Accesibilidad web
   c) Accesibilidad al contenido audiovisual en la Web.
2) Llevar a cabo un estudio extenso de la especificación HTML5
3) Elaborar un análisis de la accesibilidad en relación al nuevo estándar HTML5, focalizado en el contenido audiovisual
4) Realizar un caso de prueba para probar los resultados obtenidos tras investigar en accesibilidad a través de HTML5 en el contenido audiovisual en la Web. Desarrollo de una interface en HTML5
   a) Diseño y desarrollo de una interfaz implementada en HTML5 para la reproducción de contenido audiovisual accesible: Análisis de requisitos, diseño e implementación.
   b) Validación de dicho interfaz. Llevar a cabo pruebas para comprobar su funcionamiento en distintos agentes de usuario web. Valoración del nivel de implementación.
   c) Discusión de los datos.
5) Conclusiones y líneas futuras

1.4. Estructura de la memoria

En este documento se han distribuido los contenidos de la siguiente manera:

En este capítulo uno se ha introducido el tema principal del proyecto, presentando la motivación que ha llevado a la realización de este proyecto fin de carrera, además de indicar los objetivos que se han perseguido.

El estado de la cuestión se incluye en los capítulos dos, tres y cuatro. En el capítulo dos se presenta el lenguaje de marcado en la Web en su nueva versión HTML5. Pasando a presentar en el capítulo tres distinta información relativa a la accesibilidad web como estándares y un estudio específico de la accesibilidad a los contenidos audiovisuales en la Web. En el capítulo cuatro se analiza el estándar HTML5 en relación a los elementos destinados a mejorar la accesibilidad en la Web.

Como caso práctico para aplicar todos los conceptos vistos de manera teórica en los anteriores capítulos, se presenta en el capítulo cinco una interfaz implementada en HTML 5 para reproducir de manera accesible contenido audiovisual como vídeo en la Web.

En el capítulo seis se incluye un presupuesto correspondiente a la realización de este proyecto. Finalizaremos con unas conclusiones y líneas futuras en el capítulo siete.

Capítulo 2

2. Introducción a HTML5

En este capítulo se presenta una introducción al estado de la cuestión en el que se encuentran la tecnología base que se va a tratar en este proyecto principalmente, el HTML, siendo su versión más reciente, y actualmente en borrado camino de convertirse en estándar, HTML5 [W3C, 2011 a]. Se ofrecerá una introducción de lo que fue la versión anterior HTML4.01 y sus principales diferencias con HTML5, que son varias y significativas.

Una de las diferencias que hay con este nuevo estándar es la creación de nuevas etiquetas que van a dar más significado semántico a la Web, como etiquetas específicas para diferentes zonas de un sitio web, como por ejemplo para el encabezado, menús laterales, pies de página, y cuerpo principal, todas ellas con una etiqueta especialmente creada para la ocasión. También se debe comentar las posibles futuras aplicaciones que tendrá HTML5, como la Web semántica, que gracias a estas nuevas etiquetas se dará un mayor significado a la Web y dotará a los motores de búsqueda una mayor facilidad a la hora de encontrar resultados más ajustados a lo que el usuario demande.

Una de las características más esperadas y que más expectación ha dado es la estandarización en la forma de incluir vídeo y audio en la Web, que permita la inclusión de estos elementos multimedia sin necesidad de ninguna aplicación externa, lo que facilita enormemente un acceso más accesible y usable a estos contenidos en la Web, y que más adelante se verá con más detalle.

Otro importante factor a considerar es cómo se está implementando el HTML5 en los distintos agentes de usuario web (navegadores), ya que este software es una pieza fundamental para un correcto acceso de un sitio web. Se podrá observar mediante un estudio que se incluye, el nivel de aceptación de los navegadores con el nuevo estándar HTML5.

Otra de las características más interesantes que incorpora HTML5 según objetivos del W3C es la posibilidad de incluir subtítulos a nivel de código, para facilitar al desarrollador la introducción de estos recursos. Actualmente esta funcionalidad no ha sido implementada, pero se espera que un futuro se pueda contar con este esperado recurso. Esta funcionalidad es uno de los pilares fundamentales para otorgar de accesibilidad al contenido multimedia, por lo que mientras tanto, se puede aportar esta funcionalidad mediante otras tecnologías como JavaScript

o Flash, tal como se ha tenido que hacer en este proyecto con la utilización de tecnología JavaScript.

2.1. *HyperText Markup Language* 5 (HTML5)

HTML5 es la nueva versión del lenguaje de marcado que se usa para estructurar páginas web, que actualmente está en desarrollo y no está terminado y con el paso del tiempo contaremos gracias a él con características nuevas y modificaciones que mejorará significativamente este nuevo estándar.

En esta sección se presenta una breve introducción de las diferencias entre la anterior versión HTML4.01 y la nueva de HTML5, para a continuación explicar en profundidad este nuevo estándar.

2.2. Estándares de la Web. HTML4.X vs HTML5

HTML4 [W3C, 1998] y las diferentes versiones de la misma familia, ha sido un lenguaje muy longevo que ha tenido que adaptarse y renovarse, para mostrar el contenido de las páginas web tal y como lo conocemos ahora. Está claro que todo lo que podemos encontrar en una página web hoy en día, no estaba hace 10 o 15 años, cuando Internet no estaba tan extendido y no se sabía cómo podría afectar a nuestras vidas hoy en día, pero lo más importante que ha dado la Web es la expansión y la posibilidad a cualquier ciudadano, de crear su propia página web para plasmar sus ideas y sus opiniones. Para ello ha tenido que adaptarse para ofrecer lo nuevo que las nuevas tecnologías son capaces de incluir en la web.

El lenguaje HTML5 [W3C, 2011 a] es la nueva versión del lenguaje con el que estará escrita la Web, que es más que una renovación del HTML4.01 [W3C, 1999 a] que con el paso del tiempo se ha quedado obsoleto.

Con este nuevo borrador se pretende crear un estándar que poco a poco sustituya al anterior, introduciendo mejoras y eliminando elementos de la anterior versión que han quedado obsoletos, y por lo tanto, ya no se usan o han quedado anticuados para dar soporte a los nuevos contenidos web que se están desarrollando y a los tiempos actuales.

Lo que HTML5 intenta aportar al desarrollo web, es un lenguaje más actual y más intuitivo a la hora de estructurar una página web, y aportar algunas mejoras como la forma de incrustar videos e imágenes. Las etiquetas han sido cambiadas por otras más representativas y no se abusa tanto de las etiquetas <DIV> para estructurar de forma lógica el código, sino que bastaría con un nombre de etiqueta específico. Es decir, en lugar de estar obligados a crear una división con el nombre como identificador, ahora bastaría con etiquetas específicas que hace que la estructura de la web sea más coherente y fácil de entender por otras personas y los navegadores podrán darle más importancia a según qué secciones de la web facilitándole además la tarea a los buscadores.

Etiquetas como <Font> y que ya no sirven en HTML5 y pasan a definirse exclusivamente con las Hojas de Estilo o CSS [W3C, 2008 a]. Otras como <b> que se siguen soportando pero que ya no se usan como antes.

Se ha orientado también para un mejor acceso multimedia. Para esto se ha creado la etiqueta <video>, que permite la reproducción de vídeo sin la necesidad de instalar ningún plug-

in, ya sea QuickTime, RealPlayer o el más extendido Flash [Adobe, 2009], aunque de momento se está buscando un códec para que esto llegue a hacerse realidad. De momento sí puede usar Flash y cualquier otro plug-in o códec para reproducir videos, pero el objetivo de crear una etiqueta <video> es unificar todos esos códec en uno sólo.

Por supuesto no va a ser fácil, porque detrás hay muchos intereses económicos interesados en que el códec más usado sea el de una determinada empresa que quiera hacerse con todo el mercado de reproducción de video. Lo que plantea HTML5 es que la reproducción de vídeo sea accesible para todo el mundo, y sobretodo cómodo al no necesitar de ninguna aplicación externa para poder funcionar. Por otro lado, esta cuestión llevaría a que el estándar HTML5 dé soporte para que se pueda cumplir con el estándar UAAG [W3C UAAG, 2005] para requisitos de accesibilidad en el agente de usuario web.

Lo más novedoso que introduce este nuevo estándar de HTML, es la posibilidad de la creación de dibujos o animaciones de cualquier tipo al servicio de la imaginación de los desarrolladores, gracias a la etiqueta <canvas>, que literalmente significa "lienzo", que abre un nuevo mundo de posibilidades a lo que a diseño gráfico de sitios web se refiere. La ventaja que esto conlleva es que no se necesita ningún tipo de plug-in para el uso de esta característica de HTML5, pero sí que el navegador sea totalmente compatible con este nuevo estándar. Sin embargo, puede producir problemas de accesibilidad al ser requerido un texto alternativo a este elemento gráfico.

En cuando a la aceptación que tienen los navegadores con HTML5 hay para todos los gustos. Hay navegadores que soportan todas sus capacidades y hay otros que no aceptan ninguna, pero es cuestión de tiempo que se acabe soportando esta nueva versión. A modo de prueba, hay una página[2] que comprueba la compatibilidad del navegador con HTML5 y saca una puntuación orientativa en medida de lo que soporte.

De los navegadores más usados y conocidos, Google con su navegador Chrome y Safari, soporta la gran mayoría de lo ofrecido por HTML5, seguido por la última versión de Firefox, que a pesar de ser uno de los navegadores más utilizados, no alcanza el nivel de Chrome y Safari por poco. Le sigue Opera que en su última versión 10.5 subsana las carencias del 10 acercándose un poco más a Firefox en el S.O. Windows, ya en Mac todavía se queda en la anterior versión.

Y en el otro lado tenemos a Internet Explorer de Microsoft que todavía en ninguna de sus versiones ni en ningún S.O., soporta HTML5. El navegador Internet Explorer representa una gran cuota de mercado en lo que al uso de navegadores en Internet se refiere, ya que viene instalado con el S.O. Windows de serie, lo que significa que puede que sea uno de los navegadores más utilizados, frente a sus competidores Firefox o Chrome, aunque han ido mermando su dominio en este sector, pero aún representa una gran cuota de mercado. En su última versión Internet Explorer 8, aún no soporta esta nueva versión de HTML pero ha sido anunciado que la siguiente versión 9 sí lo soportará, lo que será la aceptación por parte de una de las empresas más importantes del mundo como es Microsoft, del nuevo estándar HTML5 y por tanto su consolidación.

[2] www.html5test.com

- Para más información, el estado de aceptación de los navegadores se describe con más profundidad en el apartado 3.3.

2.2.1. Fragmentación entre plataformas: ¿Flash o HTML5?

Últimamente se está hablando mucho sobre si HTML5 va a ser una alternativa real a Flash en el campo del vídeo. Actualmente Flash está en prácticamente en todo elemento relacionado con Internet, ya sea multimedia(videos, animaciones, etc..) como juegos online, y presente en gran parte de los ordenadores conectados a internet, ya que es necesario para reproducir la gran mayoría de elementos multimedia y además lleva mucho más tiempo disponible que HTML5. Por otro lado, una de las empresas más en auge de los últimos años, Apple [Apple, 2011 a], ha declarado continuamente que no dará soporte a Flash en su sistema operativo iOS, dando su apoyo a HTML5 considerándolo como el futuro del vídeo en Internet. El gran problema en este sector es el códec, ya que al estar actualmente en borrador, el nuevo estándar HTML5 no estará listo en unos años, por lo que habrá que esperar para resolver este problema, mientras algunas empresas declaran su apoyo a uno o a otro decodificador-codificador de vídeo. De momento la resolución a esta batalla por el control del vídeo en Internet es incierta, ya que el proceso de sustituir a una tecnología con unas capacidades tan amplias, y además tan aceptada y extendida en el los ordenadores del mundo, conlleva un tiempo de adaptación a un nuevo estándar.

La portabilidad es una de las características que muchos desarrolladores buscan. La de programar una aplicación en un lenguaje en una plataforma y sirva para cualquiera. Esto es lo que HTML5 propone, y aunque está lejos de cumplirse, este nuevo estándar se acerca bastante a lo que los desarrolladores desean.

En los últimos meses se anunció el lanzamiento, todavía en fase de prueba experimental, de una herramienta innovadora por parte de Adobe basada en tecnología AIR con el fin de conseguir compatibilidad tanto en sistemas Windows como Mac OS X. Esta herramienta se llama Wallaby [Adobe, 2011], y permite al desarrollador la conversión de Flash a HTML5 de una forma muy sencilla gracias a una interfaz intuitiva. Con esta herramienta se permite el soporte de algunos dispositivos que antes no permitía la tecnología de Adobe como son los sistemas operativos iOS de Apple y con el fin de cumplir con todos los estándares.

2.3. Algunas características que aporta HTML5

El nuevo estándar HTML5 aporta nuevas funcionalidades, que quizá no estén disponibles actualmente pero sí se han dado los primeros pasos para incluir las siguientes nuevas características a la Web.

2.3.1. Web Semántica

La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de metadatos semánticos y ontológicos en una

infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante [W3C, 2010 a].

En esta línea, HTML5 incluye elementos que permiten añadir semántica de la página web más precisa para los buscadores, para que estos la comparen con la que el usuario quiere encontrar y así se puedan obtener resultados más relevantes.

2.3.2. Geolocalización

Una de las novedades que ofrece HTML5 son las posibilidades de geolocalización a través de un punto de acceso a internet, más comúnmente conocido como Wi-Fi, en los que su uso no está bien definido pero que con el tiempo se podrá implementar en diversos servicios en redes sociales tan de moda últimamente, o para buscar negocios rápidamente sabiendo nuestra posición física. Esta funcionalidad se implementa mediante APIs que interactúan con el navegador obteniendo su IP y a través de tecnologías como JavaScript conseguir la posición geológica del usuario. Esto acarrea problemas de seguridad y privacidad, ya que enviar la posición física de un usuario no suele ser muy seguro, pero esta funcionalidad solo se podrá aprovechar si dicho usuario da su permiso expreso para que la API pueda calcular su localización geológica.

2.3.3. Aplicaciones offline

Uno de los conceptos que renueva HTML5 es la posibilidad de trabajar con aplicaciones web pero de manera offline. Lo que esta nueva funcionalidad permite es la de acceder a aplicaciones offline, sin conexión a internet, siendo necesario previamente conectarse por primera vez a dicha aplicación o página y el navegador descargará los ficheros necesarios de la página a la que se ha accedido, abriendo la posibilidad de poder volver a acceder a esta aplicación o página web sin la necesidad de conexión a Internet.

Si bien ya en HTML4.01 hay páginas que pueden visualizarse sin estar conectados a Internet (si previamente fue descargada), el uso de aplicaciones web offline se hace mucho más complejo. HTML5 plantea una revolución en este tema. Google, el gigante de los buscadores y de las aplicaciones en línea es una de las empresas que apuesta por HTML5 y ha mostrado especial atención en esta característica, para poder brindar a los usuarios que eligen sus servicios, nuevas opciones para trabajo offline, que irán perfeccionándose a medida de que HTML5 logre mayor aceptación y compatibilidad con los navegadores.

2.4. Nivel de implementación en los navegadores de HTML5

Las Tablas 1 y 2 informan al desarrollador de los elementos y atributos presentes en dichas tablas respectivamente, que están disponibles en la nueva versión HTML5 y soportados

por los distintos agentes de usuario web o navegadores. Estas tablas se han elaborado a partir de la información proporcionada en la página[3], la cual se actualiza mensualmente o cada vez que sale una nueva versión de los navegadores de los aquí mencionados. .

Después de testear la compatibilidad con los navegadores principales y sus versiones más recientes, quedaba por comprobar esta compatibilidad en versiones Beta, no completas pero sí avanzadas de estos navegadores, y los elegidos fueron Mozilla Firefox 4.0 Beta 7 e Internet Explorer 9.0 Beta. El resultado de comprobar la compatibilidad con HTML5 en estas nuevas versiones Beta de los navegadores más populares fue excelente.

Las anteriores versiones aceptaban muy poco o nada de los nuevos elementos de HTML5, por lo que dan un cambio de rumbo importante, en cuanto a la aceptación y expansión de éste nuevo estándar. De momento, como versiones Beta que son, no tienen una compatibilidad completa pero sí conforman un aceptable número de elementos, lo suficiente como para aceptar las etiquetas o funcionalidades más esperadas de HTML5, que son <video> y <audio>.

Lo que es evidente es que HTML5 se va a imponer y es casi obligatorio que los navegadores acaben implementándolo completamente, ya que las ventajas que ofrecen con el anterior son muy significativas.

Lo que los responsables de dicha página quieren dejar claro, es que no intentan persuadir de no usar el nuevo estándar HTML5 para desarrollar páginas web, sino informarle de los posibles problemas que podrá tener a la hora de implementar unos u otros elementos.

Actualmente la versión definitiva de Mozilla Firefox 4.0, ya no es Beta, y proporciona una excelente aceptación de HTML5 siendo prácticamente igual a mejor que su anterior versión. Mientras tanto, Internet Explorer aún no ha lanzado la versión definitiva 9.0, por lo que no se puede comprobar si la aceptación de HTML5 puede rivalizar con Firefox.

Ésta tabla está actualizada a fecha de 11/03/2011:

Elementos HTML5	**Chrome 10**	**Firefox 4.0 RC**	**IE9 RC**	**Opera 11**	**Safari/ Webkit r74232**
Elemento **article**	×	✓	×	×	×
Elemento **aside**	×	✓	×	×	×
Elemento **audio**	×	✓×	✓×	✓×	×
Elemento **canvas**	×	×	✓×	×	×
Elemento **datalist**	✓×	✓×	–	✓×	–
Elemento **details**	–	–	–	–	–
Elemento **figcaption**	×	✓×	×	×	×

[3] www.html5accesibility.com

Elemento **figure**	×	✓×	×	×	×
Elemento **footer**	×	✓×	×	×	×
Elemento **header**	×	✓×	×	×	×
Elemento **hgroup**	–	–	–	–	–
Input **color**	–	–	–	✓×	–
Input **date**	✓×	–	–	×	–
Input **date an time**	✓×	–	–	×	–
Input **local date and time**	✓×	–	–	×	–
Input **e-mail**	–	–	–	×	–
Input **month**	✓×	–	–	×	–
Input **number**	✓×	–	–	✓×	–
Input **range**	✓×	–	–	✓×	×
Input **search**	✓	✓	✓	×	✓
Input telephone	×	–	–	×	–
Input **time**	✓×	–	–	✓×	–
Input **url**	–	–	–	×	–
Input **week**	✓×	–	–	✓×	–
Menú > context menú	–	–	–	–	–
Menú > list	–	–	–	–	–
Menú > toolbar	–	–	–	–	–
Elemento **meter**	×	–	–	–	×
Elemento **nav**	×	✓×	×	×	×
Elemento **output**	–	–	–	×	–
Elemento **progress**	×	–	–	×	×
Elemento **section**	×	✓×	×	×	×
Elemento **summary**	–	–	–	–	–
Elemento **video**	×	✓×	✓×	✓×	×

Tabla 1: Índice de implementación de elementos HTML5 por navegadores

Atributos HTML5	**Chrome 10**	**Firefox 4.0 b10**	**IE9 Beta**	**Opera 11**	**Safari/ Webkit r74232**
Atributo **hidden**	–	✓	–	–	✓
Atributo **required**	–	✓	–	×	–
Atributo **placeholder**	×	✓×	–	×	✓×

Tabla 2: Índice de implementación de atributos HTML5 por navegadores

Leyenda de Tabla 1 y 2:

- ✗ "no soportado" significa que la característica está implementada por el navegador, pero el soporte de accesibilidad requerido no está implementado.
- ✓✗ "parcialmente soportado", significa que la característica está implementada pero el soporte de accesibilidad está parcialmente implementado.
- — "no implementado", significa que la característica todavía no está implementada por el navegador.
- ✓ "soportado" , significa que el navegador expone información de la característica a través de un acceso a la API y si la función es interactiva, puede funcionar con una amplia gama de tipos de dispositivos de entrada.

2.5. Especificación borrador HTML5.0

Esta nueva versión de HTML no es un lenguaje totalmente nuevo, sino que utiliza muchos elementos y conserva prácticamente idéntica la estructura de su anterior versión, pero con cambios sustanciales que le hacen ser más intuitivo y fácil a la hora de estructurar una página web.

Para elaborar un documento en HTML5 lo primero que se debe situar al principio es un DOCTYPE nuevo. Es muy similar a la anterior versión 4/4.01, pero esta vez es infinitamente más simple, bastaría con el código <!DOCTYPE html> siendo indiferente si se escribe en mayúsculas o minúsculas, lo que ahorra la memorización del DOCTYPE antiguo o tener que abrir un documento previo y copiarlo, y la gran ventaja es que es compatible con las anteriores versiones de HTML y XHTML, con ello se simplifica muchísimo la posible inclusión de estos estándares.

Antes de HTML5, un DOCTYPE que activa el modo estándar en todos los navegadores modernos lo podemos ver en la figura 1:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Figura 1: Doctype HTML4.01

Con HTML5 podemos conservar el anterior, pero seguramente que casi siempre se optará por el siguiente, ya que la simplicidad y la comodidad con el anterior es evidente, como muestra la figura 2:

```
<!DOCTYPE html>
```

Figura 2: Doctype HTML5

El elemento raíz de todo documento es la etiqueta <html> y ha venido acumulando elementos redundantes, quedando así en XHTML y mostrado en la figura 3:

```
<html xmlns="http://www.w3.org/1999/xhtml"
lang="en"
xml:lang="en">
```

Figura 3: Elemento raíz HTML4.01

- Pero ahora todos los elementos de HTML5 están en el mismo *namespace* y no hace falta repetir el idioma, mostrado en la figura 4:

```
<html lang="en">
```

Figura 4: Elemento raíz HTML5

2.5.1. Nuevas etiquetas

Como se ha venido introduciendo anteriormente una de las interesantes aportaciones es la facilidad al diseñar la estructura de un sitio web. Lo que hace esto más sencillo, es evitar el abuso de las etiquetas <DIV> para estructurar de forma lógica el contenido, y en su lugar se han creado etiquetas específicas para cada situación. [W3C, 2011], [W3Schools, 2010 d].

Las nuevas etiquetas son las siguientes:

- **<article>:** representa un componente de la página o sitio web, con la intención de que pueda ser reutilizado y repetido. Sería un elemento perfecto para foros o blogs que cada día se añada información nueva o noticias.
- **<section>:** representa una sección genérica de un documento o una aplicación. Podría agrupar contenido de un tema específico dentro de un artículo en que se pueda introducir, y que tenga sentido.
- **<aside>:** representa una sección de la página que abarca un contenido tangencialmente relacionado con el contenido que lo rodea, por lo que se le puede considerar un contenido independiente.
- **<header>**: representa un grupo de artículos introductorios o de navegación. Normalmente se relaciona con el encabezado a una sección de la página, mediante la inclusión de etiquetas de título (h1-h6).
- **<nav>**: representa una sección de una página que es un link a otras páginas o a partes dentro de la página: una sección con links de navegación.
- **<footer>**: representa el pié de una sección, con información acerca de la página/sección que poco tiene que ver con el contenido de la página, como el autor, el copyright o el año.

En la figura 5, podemos ver gráficamente cómo quedarían estas nuevas etiquetas en una página web esquemáticamente, comparándolo con la versión de HTML anterior y HTML5.

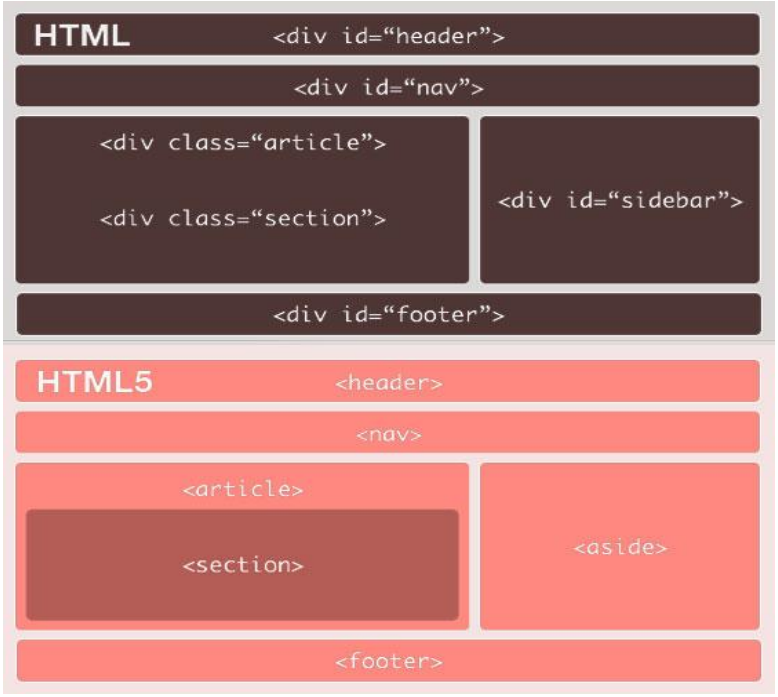

Figura 5: Esquema de etiquetas HTML5

Otras etiquetas nuevas:

- **<hgroup>:** representa el encabezamiento de una sección. Se usa para agrupar los elementos h1-h6 cuando el encabezamiento tiene múltiples niveles, como sub-encabezados o títulos alternativos.
- **<time>:** permite introducir la hora y fecha cuando se modifica o introduce información en la web, lo que permite saber al usuario cuando ha sido creado o modificado algo, y es muy útil a la hora de buscar ya que es más fácil encontrar lo más nuevo o lo más antiguo.
- **<progress>:** lo usaremos cuando queramos representar el estado de cierto proceso, muy útil para cargas y descargas de archivos.
- **<meter>:** se utilizará para indicar ciertas medidas dependiendo de los atributos, por ejemplo el tamaño del disco usado.
- **<mark>:** se usa para indicar importancia al texto, o cuando queramos atraer la atención del usuario como si subrayamos un texto con un rotulador fluorescente.
- **<video>:** es una de las etiquetas más nuevas más esperadas y más novedosas de este nuevo estándar. Permite introducir vídeo en el sitio web sin la necesidad de instalar ningún otro códec o aplicación externa. Se explicará con más detalle más adelante.
- **<audio>:** permite introducir pistas de audio con más facilidad que en la versión anterior de HTML.
- **<canvas>:** una de las grandes novedades junto con la etiqueta <video> que permite "dibujar" directamente en la pantalla del navegador, dando libertad tanto a los desarrolladores como a los usuarios en crear todo tipo de dibujos o decoraciones para su página web.
- **<details>:** permite la posibilidad de dar información adicional sobre algún elemento si el usuario lo desea
- **<dialog>:** lo usaremos para representar una conversación entre varias personas.

- **<figure>:** se utilizará para representar una imagen, o contenido relacionado entre sí o grupo de elementos (texto o imágenes).
- **<legend>:** es un elemento que se puede usar dentro de <figure>, que puede estar o no, y sirve como leyenda del contenido de su elemento padre.
- **<fieldset>:** sirve para dibujar un rectángulo y agrupar el contenido dentro de ella. En este caso, si introducimos el elemento <legend> dentro de este otro elemento, podremos escribir el título de este contenido en el mismo rectángulo.
- **<ruby>:** define anotaciones en letras y caracteres chinos.
- **<rt>:** se usa dentro de la etiqueta <ruby>. Define una explicación o pronunciación de caracteres chuinos.
- **<rp>**: es similar a <rt> pero define lo que los navegadores tienen que enseñar si no soportan la etiqueta <ruby>.
- **<keygen>**: este elemento básicamente lo que hace es generar una clave privada y una pública para que el navegador y el servidor puedan comunicarse.
- **<summary>**: define el encabezamiento de un elemento <details>.
- **<wbr>**: sirve para evitar que el navegador no rompa palabras que puedan ser conflictivas o si son demasiado largas.

En el apartado de formularios, se han añadido nuevos elementos que son permitidos en el campo "type" por lo que, ahora los datos que se introduzcan en cada formulario serán más específicos y más reconocibles que antes. [W3Schools, 2010 c]

Los nuevos elementos son los siguientes:

- **<input type="search">:** para cajas de búsqueda.
- **<input type="number">:** para sumar o restar números.
- **<input type="range">:** para seleccionar un valor entre dos valores predeterminados.
- **<input type="color">:** seleccionar un color.
- **<input type="tel">:** útil para números telefónicos.
- **<input type="url">:** útil para direcciones web.
- **<input type="email">:** para direcciones de email.
- **<input type="date">:** para seleccionar un día del calendario.
- <input type="month">: meses.
- <input type="week">: semanas.
- <input type="time">: fechas.
- **<input type="datetime">:** para una fecha exacta, absoluta y tiempo.
- **<input type="datetime-local">:** para fechas locales y frecuencia.

Estos nuevos elementos vienen a sustituir al clásico "text" que poníamos en type para todo o casi todo que quisiésemos poner en un formulario. Lo que hace esto es mejorar la búsqueda a la hora de rellenar un campo, por ejemplo, si estamos rellenando un campo de tipo url, al empezar a rellenar nos saldrá una lista de las url que ya hemos visitado y así facilitar al buscador desde qué campo selecciona la lista.

2.5.2. Nuevos atributos:

Se han introducido nuevos atributos y características con los que dotar a los elementos del HTML5, algunos de ellos orientados a la accesibilidad:

- **Ping:** Este atributo contiene una lista de URLs, las cuáles serán *llamadas* cuando un usuario haga click en ese enlace, dentro del elemento a. Por ejemplo, un uso práctico sería para estadísticas.
- **Target**: Especifica el destino donde se va a abrir un enlace distinto a la ventana actual, y aplicable en <a>, <area>, <base>, <form> y <link>.
- **Autofocus**: Destinado para indicar el elemento <input /> (no hidden), <select />, <textarea /> o <button /> que ha de coger el foco al cargar la página. Este atributo permite una mayor accesibilidad, ya que da la posibilidad de que cuando se carga la página, el foco se sitúe en un determinado elemento, por ejemplo, en la barra de búsqueda principal en una página de búsquedas en la Web.
- **Form**: nuevo atributo relacionado con los elementos de formularios. Indica a qué formulario pertenece un elemento y permite poner dicho elemento en cualquier parte de una página.
- **Replace**: atributo para <input /> <button /> y <form /> que le afectará cuando el contenido del elemento sufra algún cambio.
- **Data**: Para <form />, <select /> y <datalist />.
- **Required**: Para elementos <input /> (Excepto hidden e image) y <textarea />, indica que el campo es obligatorio. Muy útil para indicar sin Javascript, que un campo es obligatorio a la hora de rellenar un formulario, y aplicando una mejor accesibilidad en dichos formularios.
- **Disabled**: Para <fieldset />, permite desactivar el fieldset por completo, es decir, todos los hijos de dicho fieldest son desactivados.
- **Autocomplete, min, max, pattern, step**: Para elementos <input /> permite delimitar las posibilidades de nuestros elementos de entrada. Estos atributos permiten una mayor accesibilidad, indicando al usuario el límite de información o datos que puede introducir en dichos elementos.
- **Scoped**: Para elemento <style />, permitirá usar hojas de estilo "scoped", es decir, si añadimos este atributo en un elemento "style", podremos aplicar solo un determinado estilo a un subárbol del documento, o lo que es lo mismo, a ese elemento y a sus hijos.
- **Async**: Para el elemento <script /> el ajax hecho atributo. Con este atributo especificamos que el código interno se puede ejecutar en cualquier momento de la página, mejorando la velocidad de carga.
- **Hidden**: define la propiedad de que un elemento sea invisible pero ocupa el espacio que le correspondería.
- **Spellcheck** (corrector ortográfico): indica si el contenido de un elemento debe ser pasado por el corrector ortográfico.
- **Subject**: especifica el ítem correspondiente de un elemento.
- **Contenteditable**: indica que se trata de un área editable.
- **Contextmenu**: puede ser usado como punto de menú contextual proporcionado por el usuario.
- **Draggable**: indica que se trata de un elemento "draggable", es decir, que el usuario pueda arrastrar un elemento. Útil a la hora de indicar elementos interactivos por parte del usuario.
- **Tabindex**: indica la posición numérica a la que llegaremos pulsando la tecla TAB. Aporta una gran accesibilidad a las páginas web para navegar por las mismas a través del tabulador. Permite al desarrollador indicar el orden que quiere que siga la

navegación al pulsar la tecla TAB, muy útil para personas con dificultades para la navegación o simplemente para hacer más accesible el contenido de una página web.

2.5.3. Elementos cambiados:

En este apartado se incluyen aquellos elementos de HTML5 que son incompatibles con HTML4:

- El elemento <a /> sin href ahora creará un enlace al sitio.
- El elemento <address /> es ahora un nuevo concepto de sección.
- El elemento <b /> ahora representa un trozo de texto a ser estilizado sin ninguna importancia.
- Para elementos <label /> el navegador no debe mover el foco desde la etiqueta al control a menos que el comportamiento sea estándar para el interfaz utilizado en la plataforma.
- <menu /> ha sido redefinido para ser usado con los actuales menús.
- El elemento <small /> ahora representa una impresión pequeña.
- El elemento definitivamente representa el énfasis puesto en trozo de nuestro texto.

2.5.4. Elementos eliminados:

En la nueva versión, algunos de los elementos anteriormente desaprovechados pasan a ser eliminados definitivamente.

- acronym
- applet
- basefont
- big
- center
- dir
- font
- frame
- frameset
- isindex
- noframes
- noscript (solo en XHTML5)
- s
- strike
- tt
- u

2.5.5. Atributos eliminados:

A continuación se muestran los atributos que ya no se pueden usar en el nuevo estándar:

- rev y charset en <link /> y <a />
- target en <link />
- nohref en <area />
- profile en <head />
- version en <html />
- name en <map />
- scheme en <meta />
- archive, classid, codetype, declare y standby en <object />
- valuetype en <param />
- charset en <script />
- summary en <table />
- header, axis y abbr en <td /> y <th />

2.5.6. Nuevos eventos

En HTML4 se añadió la posibilidad que se activen una serie de acciones en un navegador, como cuando un usuario hace clic en un elemento mediante JavaScript.

A continuación se presentan los nuevos eventos de atributos que se pueden insertar en HTML5 para definir acciones de sucesos [W3Schools, 2010 b].

2.5.7. Eventos de ventana

Eventos relacionados con acciones que se realicen en la ventana de los navegadores:

- **Onafterprint**: script de que se ejecute después de que el documento se imprima.
- **Onbeforeprint**: script de que se ejecute antes de que el documento se imprima.
- **Onbeforeonload**: Secuencia de comandos para ejecutar antes que se cargue el documento
- **Onerror**: Script para ejecutarse cuando se produzca un error.
- **Onhaschange**: Script para ejecutarse cuando el documento haya cambiado.
- **OnMessage**: Script para ejecutarse cuando el mensaje se active.
- **Onoffline**: Script para ejecutarse cuando el documento esté desconectado.
- **Ononline**: Script para ejecutarse cuando el documento se ponga en línea.
- **Onpagehide**: Secuencia de comandos que se ejecutan cuando la ventana esté oculta.
- **Onpageshow**: Secuencia de comandos que se ejecutan cuando la ventana se haga visible.
- **Onpopstate**: Secuencia de comandos que se ejecutan cada vez que la entrada al historial cambia.
- **Onredo**: Script para ejecutarse cuando el documento realiza una “rehacer”.
- **Onresize**: Secuencia de comandos que se ejecutan cuando la ventana cambia de tamaño.
- **Onstorage**: Script para ejecutarse cuando se carga un documento.
- **Onundo**: Script para ejecutarse cuando un documento realiza una “deshacer”.
- **Onunload**: Script para ejecutarse cuando el usuario abandona el documento.

2.5.8. Eventos de formulario

Eventos relacionados con elementos de formularios en los que el usuario tenga que introducir algún tipo de información en un cuadro de texto o similar:

- **Oncontextmenu**: Script que se ejecute cuando un menú contextual se activa.
- **Onformchange**: Script para ejecutarse cuando una cambio de forma.
- **Onforminput**: Script para ejecutarse cuando un formulario de entrada del usuario se ejecuta.
- **Oninput**: Script para ejecutarse cuando un elemento recibe la entrada del usuario.
- **Oninvalid**: Script para ejecutarse cuando un elemento no es válido.

2.5.9. Eventos del ratón

Eventos activados por un ratón o acciones similares de usuario. Aplicable a todos los elementos de HTML5 [W3Schools, 2010 a].

- **Ondrag**: Script para ejecutarse cuando un elemento se arrastra.
- **Ondragend**: Secuencia de comandos que se ejecutan al final de una operación de arrastre.
- **OnDragEnter**: Script para ejecutarse cuando un elemento ha sido arrastrado a un destino válido.
- **OnDragLeave**: Script para ejecutarse cuando un elemento deja un destino válido.
- **OnDragOver**: Script para ejecutarse cuando un elemento se arrastra sobre un destino válido.
- **OnDragStart**: Script se ejecute en el inicio de una operación de arrastre.
- **OnDrop**: Script para ejecutar cuando el elemento arrastrado se cayó.
- **OnMouseWheel**: Secuencia de comandos que se ejecutan cuando la rueda del ratón se está moviendo.
- **OnScroll**: Secuencia de comandos que se ejecutan cuando la barra de desplazamiento de un elemento se desplaza.

2.5.10. Eventos multimedia

Eventos activados en elementos multimedia como <video> y <audio>, para un mayor control por parte del desarrollador web a la hora de implementar estas nuevas funcionalidades:

Aplicable a todos los elementos de HTML5, pero es más común en los elementos multimedia [W3Schools, 2010 a].

- **oncanplay**: Secuencia de comandos que se ejecutan cuando los elementos multimedia se puedan iniciar, pero puede parar para un almacenamiento temporal.
- **Oncanplaythrough**: Secuencia de comandos que se ejecutan cuando los elementos multimedia se puedan reproducir hasta el final, sin parar para un almacenamiento temporal
- **Ondurationchange**: Secuencia de comandos que se ejecutan cuando la longitud de los elementos multimedia cambian.

- **Onemptied**: Script para ejecutarse cuando un recurso de un elemento multimedia de repente se desconecta (errores de red, errores de carga, etc...)
- **Onended**: Secuencia de comandos que se ejecutan cuando los elementos multimedia llegan al final.
- **Onerror**: Secuencia de comandos que se ejecutan cuando se produce un error durante la carga de un elemento.
- **Onloadeddata**: Script para ejecutarse cuando los datos se han cargado
- **Onloadedmetadata**: Secuencia de comandos que se ejecutan cuando la duración y los datos de otros elementos multimedia se cargan.
- **onLoadStart**: Script para ejecutarse cuando el navegador empieza a cargar los datos de los elementos multimedia.
- **OnPause**: Script para ejecutarse cuando los datos de los elementos multimedia se pausan.
- **onplay**: Script para ejecutarse cuando los datos de los elementos multimedia van a empezar a reproducirse.
- **Onplaying**: Script para ejecutarse cuando los datos de los elementos multimedia han empezar a reproducirse.
- **OnProgress**: Secuencia de comandos que se ejecutan cuando el navegador está obteniendo los datos de los elementos multimedia.
- **Onratechange**: Secuencia de comandos que se ejecutan cuando la tasa de datos de los elementos multimedia de reproducción han cambiado.
- **Onreadystatechange**: Script que se ejecute cuando cambia al estado listo.
- **Onseeked**: Script para ejecutar cuando el elemento de la búsqueda de un atributo de los elementos multimedia ya no son ciertos, y la búsqueda ha terminado.
- **Onseeking**: Script para ejecutar cuando el elemento de la búsqueda de un atributo de los elementos multimedia son ciertos, y la búsqueda ha comenzado.
- **Onstalled**: Secuencia de comandos que se ejecutan cuando se produce un error al ir a buscar datos de los elementos multimedia.
- **Onsuspend**: Script para ejecutarse cuando el navegador ha estado buscando datos de los medios multimedia, pero se detuvo antes de que el archivo de los elementos multimedia fuesen encontrados.
- **Ontimeupdate**: Secuencia de comandos que se ejecutan cuando los elementos multimedia cambia su posición de reproducción.
- **Onvolumechange**: Secuencia de comandos que se ejecutan cuando se cambia el volumen de los elementos multimedia, también cuando el volumen esté ajustado en "silencio".
- **Onwaiting**: Secuencia de comandos que se ejecutan cuando los elementos multimedia han dejado de reproducirse, pero se espera que reanude.

2.5.11. Extensión de HTMLDocument

HTML5 también ha modificado el elemento padre de *Document Object Model* (DOM) Level 2. En él encontramos una serie de mejoras y otras que finalmente se hacen estándares, es decir, durante el tiempo que éste nuevo estándar se mantenga como borrador, estará en continuo cambio y se incluirán nuevas modificaciones, por lo que algunas serán imprescindibles para el correcto uso de HTML5 debido a su importancia y además también serán aplicables a través de JavaScript.

- **getElementsByClassName(),** para seleccionar elementos por el atributo class. Ya lo comentamos hace tiempo y vimos que las diferencias a nivel de tiempo de respuesta eran más que satisfactorias.
- **innerHTML**, aunque prácticamente se usa en todas, o casi todas, las aplicaciones web existentes, por fin será reconocido como estándar en la especificación. Además aprovechando su inserción se posibilita su uso en el elemento padre.
- **activeElement, hasFocus()**, nos permitirá conocer el elemento activo en tiempo real y el que tenga el foco.
- **getSelection(),** devuelve un objeto con la selección actual.
- **designMode y execCommand(),** muy usados para editar documentos.

2.5.12. Extensiones de Elementos HTMLElement.

A nivel de elementos de HTML5, el DOM también ha sufrido una serie de cambios:

- **getElementsByClassName(),** nos permite seleccionar los hijos de cualquier objeto que contengan una clase determinada.
- **innerHTML,** nos permite leer/modificar el contenido de un nodo (al añadir crea nodos texto con etiquetas).
- **classList**, una implementación muy interesante para vivir con className que nos permite interactuar con las clases de los elementos, proporcionando métodos como has(), add(), remove() y toggle() con los que podremos trabajar con las clases de nuestros elementos.
- **relList,** funciona de igual forma que classList, pero sobre los atributos rel de <a />, <area /> y <link />.

2.6. HTML5 y elementos relacionados con el contenido multimedia

HTML5 introduce importantes elementos relacionados con la multimedia, foco de este proyecto, por ello que se dedique este apartado a profundizar en estos elementos.

Hasta ahora, había que escribir varias líneas de código para introducir un vídeo o un audio, usar otro lenguaje o incluso incorporar otras tecnologías para incluir elementos multimedia en las páginas web. Pero con la llegada de este nuevo estándar, se quiere simplificar muchísimo la forma de implementar estos contenidos, ya que lo que se busca es la simplicidad obteniendo prácticamente el mismo resultado. Sin embargo, se siguen encontrando algunos inconvenientes, como por ejemplo, el formato que se utilizará en los videos, y que a continuación se explicará.

2.6.1. Etiqueta <video>

Ésta es la etiqueta que más está dando y dará que hablar, ya que va a suponer un antes y un después en la forma de mostrar páginas web en Internet. El elemento que se marca con la etiqueta o *tag* <video> es sin duda el atractivo de este nuevo estándar y es que su llegada puede revolucionar el mundo multimedia en Internet.

Hasta ahora, nunca ha habido un estándar para mostrar videos en páginas web, por una parte porque en HTML4 no había ninguna etiqueta que englobara a todos y al haber muchos tipos de códec de vídeo diferente, cada vez se hacía con el que mejor convenía, ya sea con los codecs de QuickTime, Windows Media, Real Player, Flash u otros.. y por otra parte, porque actualmente los videos son contenido de tipo esencial en Internet, algo que antes no sucedía.

Actualmente, la mayoría de los videos son mostrados a través de la instalación de un plug-in externo, como uno de los más extendidos, el de tecnología Flash, sin embargo no todos los navegadores ni todos los reproductores usan el mismo códec.

HTML5 propone un estándar para todos los videos que se incrusten en la Web, lo cual no va a ser sencillo, ya que imponer un códec para todos y desde los actores que desarrollan los reproductores se antoja algo imposible. De cualquier manera, aún queda tiempo para que el nuevo estándar se consolide y se pueda disfrutar de todas las mejoras de esta nuevo elemento.

Actualmente, hay tres formatos de vídeo disponibles para el elemento <video>:

- Ogg[4]: archivos Ogg con códec de vídeo Theodora y códec de audio Vorbis.
- MPEG4[5]: archivos MPEG4 con códec de vídeo H.264 y códec de audio ACC.
- WebM[6]: archivos WebM con códec de vídeo VP8 y códec de audio Vorbis.

Sin dudas las mejoras son evidentes respecto a Flash comparado con H.264:

- La calidad nativa del códec H.264 es mucho mejor que Flash, que al final y al cabo, no fue inventado para reproducción de videos.
- El uso de procesador requerido para ver videos usando H.264 es notablemente inferior a ver el mismo vídeo usando el plug-in de Flash[7].

En la siguiente tabla (ver tabla 3) se muestra la relación de varios navegadores con dos de los formatos de vídeo que se emplean en el momento [W3Schools, 2010 e]. Como se puede observar el formato Ogg es el que tiene más aceptación por parte de los navegadores. Esto se debe a que el formato Ogg es libre tanto como códec como contenedor, y tiene la ventaja de que se si usara este para cualquier navegador ya no sería necesario ningún software adicional, pero esto ya es decisión de las principales empresas de Internet como Adobe.

Por otro lado, el formato WebM ya se está usando de forma masiva en YouTube por parte de Google, que ya ha comenzado la conversión de la mayoría de los videos alojados en los servidores de la famosa página de visualización de videos online.

Formato	IE8	Firefox 3.5	Firefox 10.5	Opera	Chrome 3.0	Chrome i 3.0	Safari
Ogg	No	3.5+	10.5+	5.0+	No		
MPEG4	No	No	No	5.0+	3.0+		
WebM	No	No	10.6+	6.0+	No		

Tabla 3: Formatos soportados en <video>

[4] http://www.vorbis.com/
[5] http://www.m4f.org/mpeg4/
[6] http://www.webmproject.org/
[7] http://www.adobe.com/es/

<Video> con HTML4

La forma de visualizar un vídeo en una página web, actualmente es similar a la que se muestra en la figura 6. Su implementación incluye el elemento <object> del estándar HTML. Sin embargo, el elemento <object> tiene el problema de que hay navegadores que no lo interpretan bien, y por ello, es frecuente verlo implementado a través del elemento <embed>. La etiqueta <embed> no pertenece a ningún estándar, sino que fue un invento de Netscape que se generalizó en el momento de su creación. Su utilización significa ir contra los estándares de compatibilidad, aunque resulta de facto compatible con todos los navegadores ya que la práctica totalidad de los navegadores es capaz de interpretarla correctamente Si tratamos de validar una página que contenga una etiqueta <embed> el resultado será negativo ya que lo propuesto en el estándar es el uso de la etiqueta <object>, pero eliminar la etiqueta <embed> puede suponer que el código deje de funcionar en los navegadores [Moreno, L. et al., 2008 b].

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
width="425" height="344"
codebase="http://download.macromedia.com/pub/shockwave/cabs/f
lash/swflash.cab#version=6,0,40,0">
<param name="allowFullScreen" value="true" />
<param name="allowscriptaccess" value="always" />
<param name="src"
value="http://www.youtube.com/v/oHg5SJYRHA0&hl=en&fs=1&" />
<param name="allowfullscreen" value="true" />
<embed type="application/x-shockwave-flash" width="425"
height="344"
src="http://www.youtube.com/v/oHg5SJYRHA0&hl=en&fs=1&"
allowscriptaccess="always" allowfullscreen="true">
```

Figura 6: Video en HTML4

<Video> con HTML5

```
<video width="640"  height="360"
src="http://www.youtube.com/demo/google_main.mp4"  controls
autobuffer><p> Try this page in Safari  4! Or you can <a
href="http://www.youtube.com/demo/google_main.mp4">download
the  video</a> instead.</p>
</video>
```

Figura 7: Video avanzado en HTML5

Lo mostrado en la figura 7 sería un ejemplo más completo de cómo mostrar un vídeo en HTML5, pero si lo queremos simplificar, todo lo que necesitamos para mostrarlo está en la figura 8:

```
<video src=”movie.ogg” controls=”controls”>
</video>
```

Figura 8: Video simplificado en HTML5

El atributo “controls” proporciona los controles nativos que ofrece HTML5 para la etiqueta video, los cuales se tendrán en cuenta pero no serán relevantes para este proyecto. Estos

controles ofrecen los botones de play, pause y control de volumen junto con una barra de progreso. Para no ocasionar problemas con el visionado de un video, siempre es recomendable añadir atributos de altura y ancho en píxeles.

Para navegadores que todavía no soporten la etiqueta <video>, bastaría con introducir texto entre estas etiquetas, por lo que el navegador ignoraría el contenido multimedia y mostraría el texto que queramos en su lugar.

El elemento vídeo permite añadir múltiples elementos, y con éstos elementos direccionar a distintos archivos de video. El navegador usará el primer formato reconocido de entre los formatos que le introduzcamos en la etiqueta <source> por lo que usaría el que aceptase y el resto los ignoraría, como se muestra en la figura 9:

```
<video width="320" height="240" controls="controls">
   <source src="movie.ogg" type="video/ogg" />
   <source src="movie.mp4" type="video/mp4" />
   Your browser does not support the video tag.
</video>
```

Figura 9: Video múltiples formatos HTML5

Atributos para la etiqueta <video>

Estos nuevos atributos de la etiqueta <video>, se pueden incluir después de dicha etiqueta directamente, como aparece en la figura 9, para realizar lo siguiente [W3Schools, 2010 e]:

- **Autoplay**: Especifica que el vídeo comenzará a reproducirse tan pronto como esté listo
- **Controles**: Especifica que los controles se mostrará como un botón de reproducción.
- **Height(p*íxeles)*:** Especifica la altura del reproductor de vídeo
- **Loop:** Especifica que el archivo multimedia se iniciará de nuevo, una vez que haya terminado.
- **Preload:** Especifica que el vídeo se cargará en la carga de la página, y listo para reproducir. Se ignora si "autoplay" está activado.
- **Src:** Especifica la dirección URL del vídeo a reproducir.
- **Width(*píxeles):*** Especifica el ancho del reproductor de vídeo.

2.6.2. Etiqueta <audio>

Ésta etiqueta también es una de las nuevas de este nuevo estándar y tiene prácticamente las mismas propiedades que su “hermano” <video>, con sus correspondientes problemas de compatibilidad de códecs. La etiqueta <audio> viene a hacer lo mismo que <video> con los videos en internet. Ésta propone un estándar para incluir archivos de audio en las páginas web sin la necesidad de instalar ninguna otra aplicación externa, ni usar reproductor en tecnología Flash, para que los usuarios puedan escuchar sonidos, canciones, podcasts,... directamente desde la página web. Como se comentaba anteriormente con <video>, ésta vez HTML5 especifica un estándar para incluir audio, con el elemento <audio> más fácil y cómodo que con HTML4, pero con los inconvenientes de los códecs que puede soportar cada navegador.

Formato	IE8	Firefox 3.5	Opera 10.5	Chrome 3.0	Safari 3.0
Ogg Vorbis	No	Si	Si	Si	No
Mp3	No	No	No	Si	Si
Wav	No	Si	Si	No	Si

Tabla 4: Formatos soportados en <audio>

En la tabla 4 se muestra la relación de varios navegadores con tres de los formatos de audio que se emplean en el momento [W3Schools, 2010 a]. Como se puede observar el formato Ogg y Wav son de los más aceptados por los navegadores, ya que desde un principio se recomendó que se aceptará Ogg como formato estándar en HTML5, sin embargo esto cambió por desacuerdos entre compañías responsables de varios navegadores por lo que se optó, de momento, de que cada uno soporte los que crea conveniente.

<Audio> con HTML4

Con el anterior estándar HTML4, como se ve en la figura 10 no era complicado introducir una pista de audio en la web, pero se necesitaba una aplicación externa, en este caso Flash, para poder reproducirlo. En cuanto a la etiqueta, se trataba como un objeto, como la mayoría de los elementos presentes en el código HTML, con el inconveniente que no se sabía que era un archivo de audio hasta que no se abriera el código.

```
<object type="application/x-shockwave-flash"
data="player_mp3.swf" width="200" height="20">
<param name="movie" value="player_mp3.swf" />
<param name="FlashVars"
value="mp3=music.mp3&showstop=1&showinfo=1" />
</object>
```

Figura 10: Audio en HTML4.01

<Audio> con HTML5

Al igual que con <video>, para incluir un archivo de audio con HTML5 es sumamente sencillo, tal y como se muestra en la figura 11:

```
<audio src=”song.ogg” controls=”controls”>
</audio>
```

Figura 11: Audio en HTML5

De momento si queremos que nuestro archivo de audio se pueda escuchar en Firefox, Chrome, y Opera deberemos optar por usar archivos en formato Ogg. Si por el contrario queremos que se pueda escuchar en Safari, deberemos incluirlo de tipo Mp3 o Wav

Para resolver el problema de qué tipo incluir para que se escuche en cualquier navegador, optaremos por la misma solución que se menciona con el elemento <video>. Para ello, incluiremos varios enlaces de distinto tipo, y el navegador reproducirá el primer formato reconocido al igual que ocurre con la etiqueta <video>, como muestra la figura 12.

```
<audio width="320" height="240" controls="controls">
<source src="song.ogg" type="audio/ogg" />
<source src="song.mp3" type="audio/mp3" />
Tu navegador no soporta la etiqueta audio.
</audio>
```

Figura 12: Múltiples formatos de audio en HTML5

Atributos para la etiqueta <audio>

Estos nuevos atributos de la etiqueta <audio>, se pueden incluir después de ficha etiqueta directamente, como aparece en la figura 12, para realizar lo siguiente [W3Schools, 2010 a]:

- **Autoplay:** Especifica que el audio comenzará a reproducirse tan pronto como esté listo
- **Controles:** Especifica que los controles se mostrará como un botón de reproducción.
- **Loop:** Especifica que el archivo multimedia se iniciará de nuevo, una vez que haya terminado.
- **Preload:** Especifica que el audio se cargará en la carga de la página, y listo para reproducir. Se ignora si "autoplay" está activado.
- **Src:** Especifica la dirección URL del audio a reproducir.

2.6.3. Etiqueta <canvas>

Ésta etiqueta, junto con las de <video> y <audio> son las más novedosas en este nuevo estándar, pero con la gran diferencia que las anteriores ya eran conceptualmente elementos conocidos y que se usaban anteriormente.

Lo que propone esta etiqueta es algo que no se había hecho nunca antes, y es la de "dibujar" literalmente en la pantalla del navegador sin la necesidad de ninguna aplicación externa, y mediante JavaScript. Lo que se va a poder hacer con <canvas> será desde figuras geométricas, sombras, interfaces de usuarios, gráficos y tablas hasta complejos adornos para una página web y hasta juegos multimedia.

El problema de este elemento es que no es accesible directamente en el navegador, es decir, no hay ningún soporte ni alternativas directas para personas con discapacidad visual que les permita acceder a este elemento. Por lo que para otorgar a este elemento de un mínimo de accesibilidad se le debería de proporcionar al menos un texto alternativo, que le permita identificar al usuario en el momento de usar programas lectores de páginas web o similares que faciliten la accesibilidad.

Básicamente el elemento <canvas> es un área rectangular, en el que el programador controla todos y cada uno de los píxeles que lo forman, creando, modificando y añadiendo efectos para conseguir el dibujo requerido. Este elemento no genera por sí solo la habilidad de "dibujar", sino que hay que usar JavaScript para desarrollar los objetos y efectos con los que se va a componer lo que queramos dar forma.

Con el código de la figura 13, crearíamos el área rectangular donde vamos a usar el código JavaScript.

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Figura 13: Elemento Canvas

Con la función getElementById de JavaScript señalada en la figura 14, obtenemos el objeto con el id “myCanvas” para encontrar el elemento canvas.

```
var c=document.getElementById("myCanvas");
```

Figura 14: Código JavaScript para Canvas

A continuación se crea un objeto context, tal y como se muestra en la figura 15.

```
var cxt=c.getContext("2d");
```

Figura 15: Segundo código JavaScript para Canvas

El objeto getContext(“2d”) mostrado en la figura 15, está definido por sí mismo en HTML5 con muchos métodos para dibujar trazados, cajas, círculos, personajes, imágenes y mucho más.

El método fillStyle lo colorea en rojo, y el método fillRect especifica la forma, posición y tamaño. Significa que crea un rectángulo rojo en <canvas> de 150x75 empezando en la esquina superior izquierda (0,0), tal y como se muestra en la figura 16 y mostrando el resultado de la misma en la figura 17.

```
cxt.fillStyle="#FF0000";
cxt.fillRect(0,0,150,75);
```

Figura 16: Ejemplo Canvas

Figura 17: Resultado del ejemplo Canvas

2.6.4. Etiqueta track

Esta etiqueta aún en desarrollo, permite al desarrollador incluir elementos de texto de forma externa para elementos multimedia tales como videos. De momento, no representa nada por sí misma, pero se espera que sea el comienzo de la introducción de los tan esperados subtítulos en HTML5.

Según el último borrador del nuevo estándar HTML5, esta nueva etiqueta permite diferentes atributos:

- Kind: permite distintos valores relacionados con elementos multimedia como “subtitles”, “captions”, “descriptions”, “chapters”, “metadata”, y que otorgan a la etiqueta de distintas clases de información seguramente para distinguir lo que son los subtitulos al tipo de elementos multimedia con el que esté relacionado, o capítulos que contenga un vídeo o simplemente información.
- Src: como en las demás etiquetas, da la dirección donde se encuentra el elemento track el cual debe existir y contener una URL válida, tales como fichero de información como archivos XML.
- Srclang: este atributo indica el idioma de la información en el que está el elemento track, permitiendo subtitulos o la información necesaria en varios idiomas.
- Label: este atributo también indica el idioma del elemento track, pero en este caso de cara al usuario por ejemplo para que sea mostrado en su interfaz de usuario.
- Default: este atributo indica, en el caso de que haya varios elementos track relacionados con un mismo elemento multimedia, cuál de ellos va a ser el que se va a mostrar al principio, y también en el caso en el que el usuario no haya seleccionado ninguno anteriormente.

Un ejemplo extraído directamente del borrador de HTML5 es el mostrado en la figura :

```
<video src="brave.webm">
 <track kind=subtitles src=brave.en.vtt srclang=en
label="English">
 <track kind=captions src=brave.en.vtt srclang=en
label="English for the Hard of Hearing">
 <track kind=subtitles src=brave.fr.vtt srclang=fr
label="Français">
 <track kind=subtitles src=brave.de.vtt srclang=de
label="Deutsch">
</video>
```

Figura 18: Ejemplo del elemento track de HTML5

Actualmente esta nueva etiqueta no tiene un uso señalado para ninguna funcionalidad en HTML5, pero se espera que sea uno de los elementos más importante para los esperados subtítulos.

En cambio, hay una aplicación de código abierto llamada LeanBack Player [LeanBack Player, 2011], que no se apoya en ningún framework de JavaScript para funcionar, y que utiliza esta etiqueta para otorgar a un vídeo de subtítulos en varios idiomas.

Es un reproductor HTML5 que han desarrollado con la novedad de que permite la incrustación de subtítulos en los videos que queramos mostrar. La forma de hacer esto es muy sencilla, ya que se hará por medio de la propiedad “track” de HTML5.

El reproductor nos permitirá acceder a los distintos controles por medio de combinaciones de teclas de acceso rápido, así por ejemplo, para el play o para el pause se utiliza la barra espaciadora, mientras que para pararlo, se puede utilizar la "s".

Para añadir el reproductor a una página, lo primero que debemos hacer es incluir el fichero de estilo, así como los archivos de javascript que forman parte del reproductor, tal y como se observa en la figura 19.

```
<link rel="stylesheet" href="./leanbackPlayer.css"
media="screen" type="text/css" title="LeanBack Player
Style"/>
<script type="text/javascript"
src="./leanbackPlayer.js"></script>
<script type="text/javascript"
src="./leanbackPlayer_en_EN.js"></script>
```

Figura 19: Código para incluir en la cebecera el reproductor LeanBack Player HTML5

Lo siguiente es añadir el código correspondiente al reproductor, indicando la ruta hacia los videos, así como la ruta de los subtítulos tal y como se puede observar en la figura 20. También incluirá una parte correspondiente a la ejecución de un reproductor en flash, cuando el navegador no reconoce los tags de HTML5.

```
<div>
    <!-- Video element -->
    <video width="420" height="240" controls
preload="metadata" poster="./poster.jpg">
        <!-- Video sources -->
        <source src="./video.mp4" type='video/mp4;
codecs="avc1.42E01E, mp4a.40.2"'>
        <source src="./video.webm" type='video/webm;
codecs="vp8, vorbis"'>
        <source src="./video.ogv" type='video/ogg;
codecs="theora, vorbis"'>
        <!-- Subtitle sources -->
        <track enabled="true" kind="subtitles"
label="English" srclang="en" type="text/plain"
                    src="./subtitle_en.sub"></track>
        <track enabled="true" kind="subtitles" label="France"
srclang="fr" type="application/ttaf+xml"
                    src="./subtitle_tt_fr.xml"></track>
        <!-- Flash Fallback see:
https://developer.mozilla.org/En/Using_audio_and_video_in_Fir
efox#Using_Flash
                AND
http://camendesign.com/code/video_for_everybody -->
        <object name="flash_fallback" id="flash_fallback"
class="flash_fallback" width="420" height="240"
              type="application/x-shockwave-flash"
data="http://domain/player.swf">
            <param name="movie"
value="http://domain/player.swf" />
            <param name="allowfullscreen" value="true" />
            <param name="allowscriptaccess" value="always" />
            <param name="flashvars"
```

Figura 20: Código del reproductor LeanBack Player HTML5

Capítulo 3
3. Accesibilidad Web

En este capítulo se presentan los estándares de accesibilidad del *World Wide Web* (W3C) para asegurar el acceso a la Web a cualquier usuario. Sin tener en cuenta la accesibilidad, no sirve de mucho una página web con mucho contenido, ya que habrá usuarios que no podrán acceder a dichos contenidos.

El W3C contribuye con una serie de estándares a seguir por los desarrolladores de páginas web, y que se incluyen en este capítulo. Estos trabajos son unas pautas para intentar evitar que la Web sea un caos y cada uno haga las cosas como crea conveniente sin tener en mente la accesibilidad. Para ello han creado diferentes guías que permiten al desarrollador crear una página web accesible, como son: las Pautas de accesibilidad al contenido en la Web (WCAG), que explican cómo hacer que el contenido de la Web sea accesible, las Pautas de Accesibilidad para Herramientas de Autor (ATAG) que muestran cómo crear herramientas de autor tal que sean accesibles para personas con discapacidad, y las más importantes para este proyecto, las Pautas de Accesibilidad para Agentes de Usuario (UAAG) que muestran cómo hacer para que los agentes de usuario (navegadores, reproductores multimedia, etc...) sean accesibles y por último ARIA (*Accesible Rich Internet Applications*) que aporta guías para hacer accesible el contenido dinámico y los controles avanzados de interfaz desarrollados con tecnologías como Ajax y tecnologías relacionadas. Todos estos trabajos están incluidos en el W3C dentro de la Iniciativa para la Accesibilidad Web o WAI (*Web Accesibility Initiative*).

3.1. Legislación y normativa relativa a la accesibilidad

El uso equitativo de las Tecnologías de la Información y Comunicación (TIC) como Internet es un derecho para todas las personas, pero para las personas con discapacidad es además una oportunidad para poder integrarlas, favoreciendo su autonomía. Algunos países, entre ellos España, han reconocido esta necesidad y se ha regulado a través de la ley. A continuación se va a mostrar legislación, normas y directivas que son aplicables a la accesibilidad web. Estas leyes, normas y/o directivas deben tenerse en cuenta cuando se va desarrollar una aplicación web [Moreno L., 2010].

3.1.1. Marco Internacional

En cuanto a legislación a nivel internacional, se pueden considerar como punto de partida las "Normas Uniformes sobre la igualdad de oportunidades para las personas con discapacidad" [UN, 1993], aprobadas por las Naciones Unidas el 20 de diciembre de 1993, cuya finalidad es "garantizar que niñas y niños, mujeres y hombres con discapacidad, en su calidad de miembros de sus respectivas sociedades, puedan tener los mismos derechos y obligaciones que los demás". Los fundamentos políticos y morales de estas normas se encuentran en la "Carta Internacional de Derechos Humanos" [UN, 1948]. En el contenido de las normas puede leerse que "aunque no son de cumplimiento obligatorio, pueden convertirse en normas internacionales cuando las aplique un gran número de Estados con la intención de respetar una norma de derecho internacional. Llevan implícito el compromiso de los Estados de adoptar medidas para lograr la igualdad de oportunidades". Posteriormente fue aprobada en la Asamblea General de la ONU el 13 de diciembre de 2006 la Convención de Derechos de las Personas con Discapacidad, que entró en vigor el 3 de mayo de 2008, después de haber sido firmada por más de 100 países y ratificada por 20 gobiernos o parlamentos. Los legisladores de muchos países se basan en estas Normas sobre la Igualdad de Oportunidades para las Personas con Discapacidad para legislar específicamente sobre la accesibilidad en la Sociedad de la Información, como en el caso de la accesibilidad web.

En Europa existen iniciativas como el Plan de Acción eEurope 2002, eEurope 2005 e i2010 por la Comisión y Consejo de Europa. De todas ellas destaca como primer antecedente, en relación a la accesibilidad, la línea de actuación de Europa 2002 [EU, 2002] cuyo objetivo principal fue "mejorar el acceso a la Web de personas con discapacidades", en consonancia con "el principio de no-discriminación, proclamado en el Tratado de la Unión Europea", donde se estableció el final de 2001 como fecha límite para la adopción de las Pautas de la Iniciativa de Accesibilidad a la Web (WAI), WCAG 1.0.

Como marco normativo relacionado destacan las normas: ISO 9241-171:2008- Accesibilidad al Software [ISO, 2008 b], ISO 9241-20:2008- Accesibilidad en productos y servicios TIC [ISO, 2008 c] y ISO 9241-151:2008-Ergonomía de interfaces web [ISO, 2008 d].

3.1.2. Marco nacional

En España la legislación en materia de accesibilidad web cuenta con el instrumento de "Ratificación de la Convención de Derechos de las Personas con Discapacidad", dado el 21 de abril de 2008 [BOE, 2008]. Además están:

- Ley de servicios de la Sociedad de la Información y de comercio electrónico. La Ley 34/2002, de 11 de julio, de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSICE) destaca que se promoverá la adopción de normas de accesibilidad por los prestadores de servicios y los fabricantes de equipos y software, para facilitar el acceso de las personas con discapacidad o de edad avanzada a los contenidos digitales [BOE, 2002].
- Ley de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad (LIONDAU). Ley 51/2003, de 2 de diciembre, de Igualdad de Oportunidades, no Discriminación y Accesibilidad Universal de para el desarrollo de la accesibilidad [BOE, 2003].

Para organizar la puesta en marcha de la LIONDAU se consideró conveniente la elaboración de instrumentos de planificación, y durante su redacción se diseñaron dos planes: el

Plan Nacional de Accesibilidad 2004-2012 y el II Plan de Acción para las personas con discapacidad 2003-2007. En respuesta se han ido habilitando decretos como el Real Decreto 1494/2007 de 12 de noviembre el 2007, por el que se aprueba el "Reglamento sobre las condiciones básicas para el acceso de las personas con discapacidad a las tecnologías, productos y servicios relacionados con la Sociedad de la Información y medios de comunicación social" [BOE, 2007 a]. En este decreto se establece que algunos sitios web deben ser accesibles según la Norma UNE 139803:2004 [AENOR, 2004] con unos plazos definidos. 38

La norma UNE 139803:2004 recoge los requisitos que han de cumplir los contenidos disponibles en Internet y otros tipos de redes informáticas, para que puedan ser utilizados por la mayor parte de las personas. Se aplica a cualquier tipo de contenido disponible en redes informáticas, con especial énfasis en los contenidos web que son accedidos mediante navegadores de Internet. Se establecen tres Niveles de Prioridad de los puntos normativos.

Según el Real Decreto 1494/2007 a partir del 2009 los sitios web de las Administraciones Públicas y otros organismos (financiados por la Administración Pública, entidades bancarias, aseguradoras, etc.) deben ser accesibles y deberían cumplir con los requisitos de Prioridad 2 de esta norma. Además, según el decreto hay que indicar en las páginas web el grado de accesibilidad, la fecha de la revisión e incluir un sistema de contacto específico a los usuarios para transmitir las dificultades de acceso.

Muy relativa a este trabajo la norma UNE 139802:2009 (Requisitos de accesibilidad del software) [AENOR, 2009], versión oficial en español de la ISO 9241-171:2008 [10]. Incluye cuatro pautas a cumplir para que un reproductor sea considerado accesible, estas pautas son: (1) el reproductor debe permitir que el usuario pueda detener, iniciar y pausar la reproducción del vídeo, (2) es necesario permitir que el usuario pueda repetir, rebobinar, pausar, adelantar o avanzar de forma rápida una reproducción, (3) se debe permitir que el usuario pueda controlar la presentación de múltiples flujos multimedia, (4) permitir actualizar alternativas equivalentes del contenido multimedia cuando se produzca un cambio en él. Asimismo la UNE 139802:2009 incluye otros requisitos relativos al contenido alternativo subtitulado, indicando que al mostrar los subtítulos el contraste con el fondo sea suficiente, que su colocación no oculte el contenido principal a reproducir, que se puedan activar o desactivar y que permitan adaptarse a cambios en la configuración de preferencias [González M. et al, 2011].

En definitiva, existen multitud de directivas que especifican requisitos que deben cumplir las aplicaciones web en algunos ámbitos como el de la administración pública y otros. De este modo, productos que no cumplan las condiciones de accesibilidad establecidas, pueden verse rechazados en este ámbito.

3.2. Iniciativa de Accesibilidad Web (WAI)

Hablar de Accesibilidad Web es hablar de un acceso universal a la Web, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y capacidades de los usuarios.

Con esta idea de accesibilidad nace la Iniciativa de Accesibilidad Web (WAI) [W3C, 2010 c]. Se trata de una actividad desarrollada por el W3C, cuyo objetivo es facilitar el acceso de las personas con discapacidad, desarrollando pautas de accesibilidad, mejorando las herramientas para la evaluación y reparación de accesibilidad Web, llevando a cabo una labor

educativa y de concienciación en relación a la importancia del diseño accesible de páginas web, y abriendo nuevos campos en accesibilidad a través de la investigación en este área.

La idea principal radica en hacer la Web más accesible para todos los usuarios independientemente de las circunstancias y los dispositivos involucrados a la hora de acceder a la información. Partiendo de esta idea, una página accesible lo será tanto para una persona con discapacidad, como para cualquier otra persona que se encuentre bajo circunstancias externas que dificulten su acceso a la información (en caso de ruidos externos, en situaciones donde nuestra atención visual y auditiva no esté disponible: pantallas con visibilidad reducida, etc.).

3.2.1. Pautas de Accesibilidad al Contenido en la Web (WCAG)

Para hacer el contenido web accesible, se han desarrollado las denominadas Pautas de Accesibilidad al Contenido en la Web (WCAG) [W3C, 2008 d], cuya función principal es guiar el diseño de páginas web hacia un diseño accesible, reduciendo de esta forma barreras a la información. WCAG consiste en 14 pautas que proporcionan soluciones de diseño y que utilizan como ejemplo situaciones comunes en las que el diseño de una página puede producir problemas de acceso a la información. Las Pautas contienen además una serie de puntos de verificación que ayudan a detectar posibles errores.

Cada punto de verificación está asignado a uno de los tres niveles de prioridad establecidos por las pautas.

- **Prioridad 1:** son aquellos puntos que un desarrollador web tiene que cumplir ya que, de otra manera, ciertos grupos de usuarios no podrían acceder a la información del sitio web.
- **Prioridad 2:** son aquellos puntos que un desarrollador web debería cumplir ya que, si no fuese así, sería muy difícil acceder a la información para ciertos grupos de usuarios.
- **Prioridad 3:** son aquellos puntos que un desarrollador web debería cumplir ya que, de otra forma, algunos usuarios experimentarían ciertas dificultades para acceder a la información.

En función a estos puntos de verificación se establecen los niveles de conformidad:

W3C WAI-A WCAG 1.0	**Nivel de Conformidad "A":** todos los puntos de verificación de prioridad 1 se satisfacen.
W3C WAI-AA WCAG 1.0	**Nivel de Conformidad "Doble A":** todos los puntos de verificación de prioridad 1 y 2 se satisfacen.
W3C WAI-AAA WCAG 1.0	**Nivel de Conformidad "Triple A":** todos los puntos de verificación de prioridad 1,2 y 3 se satisfacen.

Tabla 5: Niveles de conformidad en WCAG 1.0

Por lo tanto en la versión 1.0 una página web tiene un nivel de adecuación A si se cumplen todos los criterios de éxito de nivel A; tiene la página web un nivel de adecuación AA si se cumplen todos los criterios de éxito nivel A y AA; y una página web puede tener nivel de adecuación AAA si se cumplen todos los criterios de nivel A, AA y AAA.

Las pautas describen cómo hacer páginas web accesibles sin sacrificar el diseño, ofreciendo esa flexibilidad que es necesaria para que la información sea accesible bajo diferentes situaciones y proporcionando métodos que permiten su transformación en páginas útiles e inteligibles.

Las Pautas de Accesibilidad de Contenidos Web 1.0 (WCAG 1.0) [W3C, 1999 b] se crearon en 1999 para guiar el diseño de las páginas web accesibles.

Las WCAG están pensadas principalmente para:

- Desarrolladores de contenido web (desarrolladores de páginas web, diseñadores de sitios web, etc.).
- Desarrolladores de herramientas de autor para la Web.
- Desarrolladores de herramientas de evaluación de accesibilidad web.

La documentación de las pautas tiene como objetivo satisfacer las necesidades de diferentes usuarios, incluyendo creadores de políticas de accesibilidad, directivos y otros.

Las WCAG 1.0 forman la versión de referencia hasta este pasado diciembre del 2008 que se publicaron las WCAG 2.0 [W3C, 2008 b] como Recomendación, después de muchos años de trabajo y borradores publicados desde 2006. Las WCAG 2.0 tienen un enfoque distinto a las WCAG 1.0 pues han sido desarrolladas para aplicarse a tecnologías actuales y emergentes y, no sólo a las tecnologías del W3C. A su vez, se busca que su utilización y compresión sea sencilla y más precisa.

Las WCAG 1.0 y las WCAG 2.0 están organizadas y estructuradas de distinta manera tal como indica la Tabla 4 [Tesis Doctoral Lourdes Moreno, Febrero 2010].

WCAG	Estructura	Puntos de Verificación, Criterios de Conformidad	Nivel de Conformidad	Documentación
1.0 (1999)	14 pautas con 65 puntos de verificación	Los Puntos de Verificación tienen asignado un Nivel de Prioridad 1, 2 o 3.	Nivel A, AA, AAA	Técnicas fundamentales Técnicas HTML Técnicas CSS
2.0 (2008)	4 principios básicos con 12 pautas que contienen un total de 61 criterios de éxito.	Los Criterios de Conformidad tienen asignados un Nivel de Conformidad A, AA ó AAA.	Nivel A, AA, AAA	Como satisfacer / Comprender Técnicas suficientes y aconsejables.

Tabla 6: Estructura de las WCAG 1.0 y WCAG 2.0

Pautas de Accesibilidad de Contenidos Web 1.0 (WCAG 1.0)

Como se ha indicado las WCAG 1.0 constan de 14 pautas que se detallan a continuación, acompañadas por una explicación:

- **Pauta 1 -** "Proporcione alternativas equivalentes para el contenido visual y auditivo".

Proporcione un contenido que, presentado al usuario, cumpla esencialmente la misma función o propósito que el contenido visual o auditivo.

- **Pauta 2 -** No se base sólo en el color.

  Asegúrese de que los textos y gráficos son comprensibles cuando se vean sin color.

- **Pauta 3 -** Utilice marcadores y hojas de estilo y hágalo apropiadamente.

Marque los documentos con los elementos estructurales apropiados. Controle la presentación con hojas de estilo en vez de con elementos y atributos de presentación.

- **Pauta 4 -** Identifique el idioma usado.

Use marcadores que faciliten la pronunciación o interpretación de texto abreviado o extranjero.

- **Pauta 5 -** Cree tablas que se transformen correctamente.

Asegure que las tablas tienen los marcadores necesarios para transformarlas mediante navegadores accesibles y otras aplicaciones de usuario.

- **Pauta 6 -** Asegúrese de que las páginas que incorporan nuevas tecnologías se transformen correctamente.

Asegúrese de que las páginas son accesibles incluso cuando no se soportan las tecnologías más modernas o éstas estén desconectadas.

- **Pauta 7 -** Asegure al usuario el control sobre los cambios de los contenidos tempo-dependientes.

Asegúrese de que los objetos o páginas que se mueven, parpadean, se desplazan o se actualizan automáticamente, puedan ser detenidos o parados.

- **Pauta 8 -** Asegure la accesibilidad directa de las interfaces de usuario incrustadas.

Asegure que la interfaz de usuario sigue los principios de un diseño accesible: funcionalidad de acceso independiente del dispositivo, teclado operable, voz automática, etc.

- **Pauta 9 -** Diseñe para la independencia del dispositivo.

Utilice características que permitan la activación de los elementos de la página a través de diversos dispositivos de entrada.

- **Pauta 10 -** Utilice soluciones provisionales.

Utilice soluciones de accesibilidad provisionales de forma que las ayudas técnicas y los antiguos navegadores operen correctamente.

- **Pauta 11 -** Utilice las tecnologías y pautas W3C.

Utilice tecnologías W3C (de acuerdo con las especificaciones) y siga las pautas de accesibilidad. Donde no sea posible utilizar una tecnología W3C, o usándola se obtengan materiales que no se transforman correctamente, proporcione una versión alternativa del contenido que sea accesible.

- **Pauta 12 -** Proporcione información de contexto y orientación.

Proporcione información de contexto y orientativa para ayudar a los usuarios a entender páginas o elementos complejos.

- **Pauta 13 -** Proporcione mecanismos claros de navegación.

Proporcione mecanismos de navegación claros y coherentes, (información orientativa, barras de navegación, un mapa del sitio, etc.) para incrementar la probabilidad de que una persona encuentre lo que está buscando en un sitio.

- **Pauta 14 -** Asegúrese de que los documentos sean claros y simples.

Asegure que los documentos son claros y simples para que puedan ser más fácilmente comprendidos.

Pautas de Accesibilidad de Contenidos Web 2.0 (WCAG 2.0)

A continuación se van a mostrar los cuatro principios generales y las doce pautas de las WCAG 2.0:

- **Principio 1: Perceptible -** La información y los componentes de la interfaz de usuario deben ser presentados a los usuarios de modo que ellos puedan percibirlos.
  - **Pauta 1.1** Alternativas textuales: Proporcionar alternativas textuales para todo contenido no textual de modo que se pueda convertir a otros formatos que las personas necesiten, tales como textos ampliados, braille, voz, símbolos o en un lenguaje más simple.
  - **Pauta 1.2** Medios tempo-dependientes: Proporcionar alternativas sincronizadas para los medios tempo-dependientes.
  - **Pauta 1.3** Adaptable: Crear contenido que pueda presentarse de diferentes formas (por ejemplo, con una disposición más simple) sin perder información o estructura.
  - **Pauta 1.4** Distinguible: Facilitar a los usuarios ver y oír el contenido, incluyendo la separación entre el primer plano y el fondo.
- **Principio 2: Operable -** Los componentes de la interfaz de usuario y la navegación deben ser operables.
  - **Pauta 2.1** Accesible por teclado: Proporcionar acceso a toda la funcionalidad mediante el teclado.
  - **Pauta 2.2** Tiempo suficiente: Proporcionar a los usuarios el tiempo suficiente para leer y usar el contenido.
  - **Pauta 2.3** Convulsiones: No diseñar contenido de un modo que se sepa podría provocar ataques, espasmos o convulsiones.
  - **Pauta 2.4** Navegable: Proporcionar medios para ayudar a los usuarios a navegar, encontrar contenido y determinar dónde se encuentran.
- **Principio 3: Comprensible -** La información y el manejo de la interfaz de usuario deben ser comprensibles.
  - **Pauta 3.1** Legible: Hacer que los contenidos textuales resulten legibles y comprensibles.
  - **Pauta 3.2** Predecible: Hacer que las páginas web aparezcan y operen de manera predecible.
  - **Pauta 3.3** Entrada de datos asistida: Ayudar a los usuarios a evitar y corregir los errores.

- **Principio 4: Robusto -** El contenido debe ser suficientemente robusto como para ser interpretado de forma fiable por una amplia variedad de aplicaciones de usuario, incluyendo las ayudas técnicas.
    - **Pauta 4.1** Compatible: Maximizar la compatibilidad con las aplicaciones de usuario actuales y futuras, incluyendo las ayudas técnicas.

3.2.2. Pautas de Accesibilidad para Herramientas de Autor (ATAG)

Las Pautas de Accesibilidad para Herramientas de Autor (ATAG) muestran cómo hacer que las herramientas de autor sean accesibles para personas con discapacidad. Estas herramientas son software que se utiliza para crear páginas y contenido web. Uno de los objetivos principales de las ATAG es definir la forma en la que las herramientas ayudan a los desarrolladores web a producir contenido web que cumpla las Pautas de Accesibilidad al Contenido en la Web [W3C ATAG, 2005].

Las ATAG están pensadas principalmente para desarrolladores de herramientas de autor. Entre estas herramientas de autor se incluyen, al ser un estándar no relativo directamente con este proyecto, no se va a profundizar en sus pautas.

3.2.3. Pautas de Accesibilidad para Agentes de Usuario (UAAG)

Los documentos de Pautas de Accesibilidad para Agentes de Usuario (UAAG) [W3C UAAG, 2005] muestran cómo hacer que los agentes de usuario sean accesibles para personas con discapacidad, en especial cómo incrementar la accesibilidad al contenido web. Entre los agentes de usuario se incluyen navegadores, reproductores multimedia y productos de apoyo, software que algunas personas con discapacidad utilizan para interactuar con los dispositivos. Por ser el estándar a seguir para hacer un reproductor accesible, sí se ha sido tenido en cuenta en este proyecto, incluyendo mucho de los requisitos en el diseño y desarrollo del interfaz caso de uso del que consta este proyecto (ver capítulo 5). Servir vídeo a través de HTML5 hace que se tenga que dar soporte en este mismo estándar o con la incorporación de otras tecnologías para poder incluir todos estos requisitos descritos en las UAAG, ya sea con la ayuda de JavaScript, Flash o CSS.

Tanto las UAAG 1.0 [W3C, 2002] como otros documentos complementarios tienen como objetivo satisfacer las necesidades de usuarios diversos, creadores de políticas, directivos y otros. Por ejemplo:

- Aquellos usuarios que deseen elegir agentes de usuario más accesibles pueden utilizar las UAAG para evaluar los agentes de usuario
- Aquellos que por otro lado quieran animar a que los desarrolladores de agentes de usuario existentes mejoren la accesibilidad en versiones futuras, pueden indicar a los proveedores de agentes de usuario como referencia las UAAG.

Las UAAG 1.0 contienen un conjunto de puntos de verificación que incluyen:

- Acceso a todo el contenido, incluyendo contenido en relación de eventos generados por el ratón o el teclado
- Control del usuario sobre la forma en que se muestra el contenido

- Control del usuario sobre la interfaz del usuario, con documentación sobre características de accesibilidad
- Interfaces de programación estándares, para permitir la interacción con tecnologías de apoyo

Desde el lanzamiento de UAAG 1.0 como Recomendación del W3C en diciembre de 2002, el Grupo de trabajo para las Pautas de Accesibilidad para agentes de usuario (UAWG) ha recibido comentarios sobre la usabilidad, comprensibilidad y aplicabilidad del conjunto de los documentos. Además, en los años transcurridos se han producido cambios y mejoras en:

- Tecnologías y técnicas que se utilizan en el contenido web.
- La funcionalidad de la tecnología de asistencia.
- Acceso a las interfaces de programación de aplicaciones (API).
- Plataformas para recibir el contenido.

Los comentarios, cambios, y la información obtenida de la evaluación de los agentes de usuario utilizando bancos de pruebas para desarrollar informes de ejecución, están impulsando el desarrollo de UAAG 2.0 [W3C, 2010 b] y es capturado como los requisitos para la misma.

El objetivo principal de UAAG 2.0 es el mismo que la versión 1.0, reducir las barreras a la accesibilidad web para personas con discapacidad.

Actualmente, las UAAG 2.0 es un documento en borrador por lo que está en continuo cambio y modificación, y como borrador que es, todavía no está reconocido por la propia W3C, y por lo tanto, no sustituye a las UAAG 1.0.

Los cambios sustanciales incluyen:

- Actualización de las definiciones de los criterios de éxito y glosario, en el que se detalla el comportamiento deseado de concentración y control del usuario sobre una indicación visual del foco.
- Actualización de los criterios de éxito para la conservación de las diferencias de tamaño relativo a las fuentes cuando su tamaño se escala. El cambio reconoce los casos de uso, cuando hay una necesidad de accesibilidad para eliminar las diferencias relativas al tamaño y que dan control al usuario.
- Actualización de los criterios de éxito en el envío de formularios para permitir al usuario redefinir los métodos abreviados de teclado para la presentación y cancelación de dichos formularios.
- Eliminar los criterios de éxito del punto 3.3.6 debido a que no era específico para usuarios con discapacidad.

Las directrices fundamentales de la versión 2.0 de las UAAG son:

- **Principio 1:** Cumplir con las especificaciones aplicables y los convenios.
- **Principio 2:** Facilitar el acceso mediante programación.
- **Principio 3:** Perceptible – La interfaz del usuario y el contenido debe ser presentado a los usuario de manera que lo puedan percibir.
- **Principio 4:** Asegurarse de que la interfaz del usuario es operable.
- **Principio 5:** Asegurarse de que la interfaz del usuario es comprensible.

Los agentes de usuario pueden reclamar la conformidad con UAAG 2.0 en uno de los tres niveles de conformidad. El nivel alcanzado depende del nivel de los criterios de éxito que se han cumplido. Los niveles de conformidad que también comparte con la versión 1.0 de las UAAG son los siguientes:

- **Conformidad con el nivel "A":** el agente de usuario cumple con todos los criterios de éxito de nivel A.
- **Conformidad con el nivel "AA":** el agente de usuario cumple con todos los criterios de éxito de nivel A y AA.
- **Conformidad con el nivel "AAA":** el agente de usuario cumple con todos los criterios de éxito.

Si se siguen los requisitos incluido en las UAAG 2.0, hay unos elementos a incorporar en la interfaz del reproductor si se quiere incluir de manera accesible un vídeo al que se acceda a través de un reproductor en una página web. Según [González M. et al, 2011] hay unos controles básicos como: reproducir o parar el vídeo, permitir cambiar el tamaño de la ventana, permitir ajustar el volumen y unos controles adicionales tales como: permitir activar o desactivar los subtítulos, permitir activar o desactivar la audiodescripción, permitir realizar búsquedas dentro de los subtítulos de la reproducción, permitir adelantar o atrasar segundos en una reproducción, permitir cambiar el tamaño, la fuente y el color del texto, permitir acceso a la documentación de ayuda donde se informe sobre los atajos de teclado existentes. La mayoría de estos controles han sido requisito en este proyecto fin de carrera

3.2.4. Web Accessibility Initiative - Accessible Rich Internet Application (WAI-ARIA)

Pero la función que tienen todas estas aplicaciones no sería posible sin una buena accesibilidad, y de esto es lo que se encarga la WAI-ARIA (*Web Accessibility Initiative - Accesible Rich Internet Application*) [W3C, 2011 d].

WAI-ARIA es una iniciativa del W3C que define cómo hacer accesibles contenidos y aplicaciones web, específicamente el contenido dinámico y los controles avanzados de interfaz desarrollados con Ajax, HTML, Javascript y sus tecnologías relacionadas.

Pretende ser una ayuda en el contenido dinámico de los interfaces actuales de las aplicaciones web, que cada día son más parecidos a los entornos de escritorio y que su propio funcionamiento puede interferir en los productos de apoyo.

Esta tecnología tiene como principal objetivo aportar información acerca de las diferentes partes que constituyen los contenidos dinámicos generados, normalmente, por medio de *scripts*. Toda esta información será utilizada por los productos de apoyo para la interacción con el usuario final.

Los documentos técnicos de WAI ARIA están desarrollados por el grupo de trabajo del W3C PFWG (*Protocols and Formats Working Group*). El W3C pone a disposición varios documentos sobre WAI-ARIA, entre ellos, destaca la especificación técnica de la W3C.

¿Cómo funciona?

WAI ARIA proporciona una serie de atributos que funcionan como identificadores de las diferentes partes de la aplicación que interactúa con el usuario. También se incluyen mapeo de controles y eventos para la accesibilidad de las APIs (*Application Programming Interfaces*).

WAI ARIA dispone de roles que describen tanto los *widgets* (componentes con funcionalidad propia de las interfaces de escritorio o web) de la aplicación como la estructura de la página web, como por ejemplo: los encabezados y las regiones. También dispone de varias propiedades como los estados de los *widgets*, las regiones activas de actualización de contenidos y sobre características *drag-and-drop*. A su vez, provee una manera de navegar mediante teclado dentro de los componentes [INTECO, 2010].

Ventajas y desventajas

Al usar esta tecnología se encuentran una serie de ventajas:

- Al añadir valor semántico al contenido y a los *widgets* se puede situar al usuario exactamente en donde está. Además de su facilidad de implementación ya que son los atributos los que los define.
- Las actualizaciones de contenido dinámico, normalmente realizado con tecnologías de *script*, son notificadas al usuario por medio de su producto de apoyo.
- Accesibilidad de los *widgets* por medio del teclado.
- Se dota de información de cómo se utiliza un *widget* y qué tipo de datos proporciona.

En cambio, se puede identificar una desventaja manifiesta, ya que al aplicar esta tecnología, los documentos web que se desarrollen no validarán tanto en HTML4 como en XHTML 1.0. El primero no soporta espacio de nombres, por lo que no se podrá incluir los atributos específicos de WAI ARIA. Con XHTML 1.0, su validación no sería posible porque no estaría incluido en el esquema que define al lenguaje.

Una posible solución sería incluir los atributos de WAI ARIA mediante DOM, es decir, incluir con JavaScript, dinámicamente, los atributos en los elementos destinados a ellos. Dicha solución haría dependiente de la tecnología *script* cualquier tipo de implementación.

Existe otra solución que podría resolver el problema: la utilización de XHTML 1.1. Este lenguaje es una especificación de lenguaje modular y extensible que permitiría personalizar el DTD incorporando el esquema de WAI ARIA [INTECO, 2010].

¿Por qué WAI-ARIA?

Tanto HTML4.01 como HTML5 fueron creados para la presentación de documentos textuales con formatos específicos para su función, como puedan ser los encabezados, marcadores de párrafos o cualquier otro tipo de elementos que tuvieran que ver con el formateo de texto. Esto es así desde que en las primeras versiones se incluyó el elemento IMG para la incorporación de elementos de imágenes dentro de los documentos.

El lenguaje fue desarrollado para un modelo de cliente-servidor, donde el usuario pedía el documento y desde un servidor se lo proporcionaba. Esta comunicación está orientada de forma secuencial, es decir, un usuario envía una petición a un servidor, normalmente de un documento HTML, y el servidor lo procesa para, finalmente, enviarlo al usuario final.

Con la evolución de la web, las páginas han ido evolucionando hacia las llamadas aplicaciones web, mucho más parecidas a los entornos de escritorio que tienen un

funcionamiento más dinámico con respecto a los contenidos. Para ello, se han utilizado tecnologías web como JavaScript, y más concretamente AJAX, que de unos años hasta hoy ha tenido un crecimiento sustancial.

También está presente el problema de la independencia de dispositivo, ya que algunos componentes de las aplicaciones web enriquecidas no son accesibles desde el teclado, orientando su uso al ratón. Esta característica entraña un grave problema de accesibilidad, dado que muchas personas con discapacidad física son incapaces de utilizar el ratón [INTECO, 2010].

Las soluciones que aporta ARIA

ARIA viene a responder a las anteriores preguntas y otras cuestiones proporcionando un marco de trabajo complementario**.**

- Estructuras más semánticas para las zonas funcionales.
- Mejora de la navegación mediante el teclado.
- Controles complejos (*widgets*) más accesibles.
- Accesibilidad para el contenido actualizado de forma dinámica.

3.2.5. Roles

Su misión es definir el papel que juegan los elementos dentro del documento web. Como se ha comentado antes, los roles dentro de WAI ARIA sirven para describir los *widgets*, así como para definir elementos propios de las páginas web. Estos atributos proporcionan información a los productos de apoyo sobre los componentes *widgets*.

Es evidente que, por definición, las etiquetas de HTML tienen un rol predefinido, el mismo que el propio elemento indica. Un rol de WAI ARIA ayuda a concretar la funcionalidad de la etiqueta en la que va definida y se podría afirmar que la sustituye, aunque en nuestro caso, HTML5 dispone de etiquetas mejor definidas semánticamente que su antecesor.

Los roles *Landmark* son un tipo que permite dotar de un significado a ciertas regiones que tienen una funcionalidad o propósito bien marcado y relevante para el uso de cualquier usuario. Las regiones destinadas a la navegación o a la búsqueda de términos pueden ser consideradas como *Landmarks.* Algunos tipos de roles *Landmarks* son:

- Article
- Banner
- Complementary
- Contentinfo
- Main
- Navigation
- Search
- Form

3.2.6. Estados y propiedades

WAI-ARIA define diferentes tipos de propiedades, los cuales determinan las características y los valores de cada elemento. Las propiedades pretenden ser una guía a los

productos de apoyo que proporciona información de cómo interactuar con ciertos *widgets* que se encuentran en una página web [INTECO, 2010].

A diferencia de los roles, en el que sólo existe un atributo *role* y varios valores (*navigation*, *menu*, *search*, etc…), los atributos de los estados y propiedades pueden ser varios, y cada uno puede aceptar varios valores.

Existen estados y propiedades de carácter global, es decir, pueden ser utilizados independientemente del elemento o del tipo de rol que tenga. Pero en la mayoría de los casos la clasificación de los estados y propiedades viene distribuida de la siguiente forma:

- Atributos *Widget*
- Atributos de regiones activas
- Atributos de Drag and Drop
- Atributos de relaciones

3.3. Accesibilidad al contenido multimedia en la Web

En esta sección se va a mostrar cómo ofrecer accesibilidad a los contenidos audiovisuales y multimedia en la Web, siguiendo las recomendaciones del estándar WCAG. Destacar que además de esta opción, hay soluciones tecnológicas de gran interés en el área de la multimedia al margen de las tecnologías W3C como puede ser la aplicación comercial Flash de Adobe [Adobe, 2008] para producir contenidos interactivos y distintos programas de presentación multimedia. La información empleada para la realización de esta apartado, se ha extraído del libro "Accesibilidad a los contenidos audiovisuales en la web: Una panorámica sobre legislación, tecnologías y estándares", en el que se recoge las recomendaciones de la nueva versión del estándar WCAG 2.0 para los contenidos multimedia y audiovisuales en la web.

Para proveer de accesibilidad a los contenidos audiovisuales, tal como se vio en el apartado 3.2 relativa a la iniciativa WAI que incluyen las WCAG y UAAG, distintos componentes de desarrollo web e interacción deben trabajar conjuntamente para conseguirlo. A esta situación se la ha denominado "la cadena de la accesibilidad de los contenidos audiovisuales en la Web". Esta cadena consta de los siguientes eslabones a tener en cuenta [Moreno L. et al., 2008 a]:

1. Contenido en sí: el contenido debe ser accesible.
2. Cómo llegar al contenido: el acceso a ese contenido debe ser accesible.
3. Visualización del contenido: hay que ofrecer alternativas atendiendo a preferencias del usuario, y la interacción del usuario al contenido debe ser usable.

Como primer eslabón de la cadena está el conseguir que el contenido en sí mismo sea accesible, lo que significa proveer de alternativas sincronizadas como el subtitulado, la audiodescripción o la trascripción, entre otros, siguiendo las WCAG.

Continuando con la cadena se considerará el siguiente eslabón: la forma de incluir el vídeo para reproducirlo en la Web, ya que aun siendo accesible el contenido audiovisual, cómo se implemente este acceso puede llegar a provocar barreras de accesibilidad. Como alternativas para ofrecer el contenido nos encontramos con la descarga directa, descarga progresiva o falso *streaming*, o con la modalidad de difusión por *streaming*. Cada una de estas opciones conlleva

formas distintas de implementación en una página web, pero todas las opciones deben seguir las WCAG, que se hablará en el siguiente

En el último eslabón de la cadena ante la diversidad de reproductores, tipos de conexiones, etc. es conveniente ofrecer alternativas atendiendo a preferencias del usuario. Pero el acceso del usuario al contenido debe ser, además de posible, intuitivo en la interacción, por lo que hay que ofrecer usabilidad, aportando accesibilidad en cómo mostrar la información de acceso y control al usuario.

3.3.1. Primer eslabón: el contenido deber ser accesible.

La accesibilidad a los contenidos audiovisuales se trata en las WCAG. En la versión WCAG 1.0 se tratan de forma muy general y poco precisa. Se habla de descripción auditiva y de subtitulado alternativo a la banda visual. En cambio, la WCAG 2.0 es más exigente y pretende ser más concreta.

Contiene más puntos a revisar: se traduce en nueve criterios de éxito, en comparación a sólo los dos puntos de verificación que hay actualmente, y se distingue entre multimedia pregrabada y multimedia en directo para el subtitulado, audiodescripción y audiodescripción extendida. Además, se considera como nuevo contenido alternativo la lengua de signos. En las WCAG 2.0 hay que ofrecer estos contenidos alternativos para obtener distintos niveles de conformidad tal como se indica en la tabla 7.

1.2.2	Captions (Prerecorded)	A	1.4	1
1.2.3	Audio Description or Full Text Alternative	A	1.4	1
1.2.4	Captions (Live)	AA	1.4.	1
1.2.5	Audio Description	AA	1.3, 1.4	1
1.2.6	Sign Language	AAA	Not mapping	
1.2.7	Audio Description (Extended)	AAA	1.3, 1.4	1
1.2.8	Full Text Alternative	AAA	1.4.	1
1.2.9	Live Audio-only	AAA	1.1.	1

Tabla 7: Correspondencia de las pautas WCAG1.0 con las WCAG2.0 en relación a los contenidos audiovisuales

Las dos pautas aplicables a los contenidos audiovisuales en la Web de las WCAG 1.0 son la 1.3 y 1.4, ambas de prioridad 1, y establecen lo siguiente:

- 1.3 Hasta que las aplicaciones de usuario puedan leer en voz alta automáticamente el texto equivalente de la banda visual, proporcione una descripción auditiva de la

información importante de la banda visual de una presentación multimedia. [Prioridad 1].

- 1.4 Para toda presentación multimedia tempo dependiente (por ejemplo, una película o animación) sincronice alternativas equivalentes (por ejemplo, subtítulos o descripciones de la banda visual) con la presentación. Según esto, habría que proporcionar transcripción de todo el contenido de la banda. [Prioridad 1].

Así, hay que proporcionar contenidos alternativos al contenido audiovisual como la audiodescripción integrada en el contenido o bien, ofrecer una descripción textual o transcripción que transmita de forma completa toda la información (información de los personajes, trascripción completa, acciones, lenguaje corporal, contexto y cambios de escena, etc.). Además, hay que ofrecer subtitulado y de manera sincronizada. Este subtitulado debe transmitir toda la información de la banda auditiva.

En las WCAG 2.0, la pauta que trata los contenidos multimedia es la 1.2, pero hay que hacer referencia al criterio de éxito 1.1.1 también aplicable, que indica que para todo contenido no textual se debe proporcionar al usuario una alternativa textual equivalente.

La pauta 1.2 indica que se deben proporcionar contenidos alternativos sincronizados a los contenidos multimedia, con distintos criterios de éxito como:

- 1.2.1 Para todo contenido sólo-audio o sólo-vídeo pregrabado se debe proporcionar una alternativa equivalente que transmita de forma completa la información de la pista de audio o de vídeo respectivamente. [Nivel A].

  Así, hay que:
  - Proporcionar una descripción textual alternativa a la pista de audio que trasmita la misma información que ésta.
  - Proporcionar una descripción auditiva de los elementos claves de la pista visual que transmita la misma información que ésta. Dicha descripción debe incluir información de los personajes, trascripción completa, acciones, lenguaje corporal, contexto y cambios de escena, etc.
- 1.2.2 Se deben proporcionar subtítulos para cualquier contenido de audio pregrabado estando sincronizados además con la pista sonora. La única excepción es que dicho audio sea a su vez alternativa de otro contenido. [Nivel A]. Así, hay que:
  - Proporcionar subtítulos abiertos o cerrados. Éstos deben representar la información contenida en la pista de audio, por tanto deben recoger tanto los diálogos de los personajes como los eventos sonoros que se pudieran producir. Éstos deben estar sincronizados con la pista visual.
- 1.2.3 Se debe proporcionar audiodescripción o alternativa textual completa de la pista visual. [Nivel A].
  - Así, hay que:
  - Proporcionar una descripción auditiva de los elementos claves de la pista visual.
  - Dicha descripción auditiva debe transmitir de forma completa toda la información de la pista visual, a saber, información de los personajes, trascripción completa, acciones, lenguaje corporal, contexto y cambios de escena, etc.
  - Como alternativa a la pista sonora equivalente se puede proporcionar una descripción textual completa que pueda ser leída por un lector de pantalla. Esta

descripción textual debe transmitir la misma información que la pista sonora equivalente.
    - La pista visual debe ir acompañada por una descripción auditiva sincronizada con la pista visual y la pista sonora.
- 1.2.4 Se deben proporcionar subtítulos para cualquier contenido audio en directo. [Nivel AA] Así, hay que:
    - Proporcionar subtítulos abiertos o cerrados. Éstos deben representar la información contenida en la pista de audio y deben estar sincronizados con ésta, lo cual requerirá técnicas especiales por tratarse de audio en directo.
- 1.2.5 Se debe proporcionar audiodescripción o de la pista visual. [Nivel AA].
    - Así, hay que:
    - Proporcionar una descripción auditiva de los elementos claves de la pista visual.
    - Dicha descripción auditiva debe transmitir de forma completa toda la información de la pista visual, a saber, información de los personajes, trascripción completa, acciones, lenguaje corporal, contexto y cambios de escena, etc.
    - La descripción auditiva estará sincronizada con la pista visual y la pista sonora.
- 1.2.6 Se proporcionará un contenido alternativo en lengua de signos para toda pista sonora pregrabada. [Nivel AAA].
    - Así, hay que controlar que:
    - Dicho contenido debe estar sincronizado con la pista de audio principal.
- 1.2.7 Se debe proporcionar audiodescripción extendida. [Nivel AAA]
    - Así, hay que:
    - En el caso de que las pausas que se produzcan en la pista de audio principal no sean suficientemente largas como para incluir audiodescripción, se deberá proporcionar audiodescripción extendida como contenido alternativo.
    - Proporcionar una descripción auditiva de los elementos claves de la pista visual.
    - Dicha descripción auditiva debe transmitir de forma completa toda la información de la pista visual, a saber, información de los personajes, trascripción completa, acciones, lenguaje corporal, contexto y cambios de escena, etc..
- 1.2.8 Se debe proporcionar alternativa textual completa para cualquier contenido audiovisual pregrabado. [Nivel AAA].
    - Así, hay que:
    - Transmitir de forma completa toda la información de la pista visual, a saber, información de los personajes, trascripción completa, acciones, lenguaje corporal, contexto y cambios de escena, etc.
    - En el caso de que se produzca algún tipo de interacción con el usuario la alternativa textual contendrá un vínculo que proporcione una funcionalidad interactiva equivalente.
- 1.2.9 Se deben proporcionar contenidos alternativos para cualquier contenido audio en directo.[Nivel AAA]
    - Así, hay que:
    - Proporcionar subtítulos abiertos o cerrados. Estos deben representar la información contenida en la pista de audio y deben estar sincronizados con esta, lo cual requerirá técnicas especiales por tratarse de audio en directo.
    - La transcripción textual completa se considera una posibilidad pero serán preferibles los subtítulos.

Técnicas

Según lo visto, en este primer eslabón, para poder cumplir con el estándar WCAG 1.0, hay que cumplir con los puntos 1.3 y 1.4 de prioridad uno (es decir, para que el sitio web fuera nivel A, por ejemplo, se debería cumplir estos puntos necesariamente). Pasamos a continuación a analizar las técnicas, herramientas y guías útiles para el diseñador. Debido a que el estándar WCAG 1.0 está obsoleto respecto a la tecnología actual, nos referiremos a la documentación de las Técnicas de WCAG 2.0. Hay que destacar que respecto a las WCAG 1.0 se ha producido un cambio de enfoque en las Técnicas ofrecidas en la versión 2.0 de WCAG; además es más práctica y útil para el diseñador aportando recursos, ejemplos y procedimientos de revisión del criterio de éxito en cada caso. Así, se recomienda familiarizarse con esta documentación, y de forma más específica con las técnicas WCAG 2.0 relacionadas y Técnicas de SMIL *(Synchronized Multimedia Integration Language*) estándar de presentación multimedia desarrollado por el W3C indicadas en la segunda columna de la tabla 6, en relación a cada uno de los criterios de éxito aplicables en la accesibilidad de los contenidos audiovisuales en la Web.

Criterios de éxito del WCAG 2.0	Técnicas del WCAG 2.0 y SMIL
1.1.1 Non-text Content	G68
1.2.1 Audio-only and Video-only (Prerecorded)	G158, G159
1.2.2 Captions (Prerecorded)	G93, G87, SM11, SM12
1.2.3 Audio Description or Full Text Alternative:	G69, G78
1.2.4 Captions (Live)	G9, G93
1.2.5 Audio Description	G78, SM6, SM7
1.2.6 Sign Language	G54, G81, SM13, SM14
1.2.7 Audio Description (ext.)	G8, SM1, SM2
1.2.8 Alternative Full Text	G69, G159
1.2.9 Live Audio-only	G150, G157

Tabla 8: Técnicas del WCAG 2.0 y SMIL para la pauta 1.2

Herramientas de autor

Para elaborar contenidos alternativos se necesitan herramientas de autor. Existe un gran número de herramientas orientadas al desarrollo y soporte de material multimedia audiovisual en Web [Moreno L. *et al*, 2006]. Es posible clasificar estas herramientas a través de muchos criterios que van desde la plataforma en que están desarrolladas o el sistema operativo bajo el que funcionan, pasando por el grado de usabilidad de la herramienta o el grado de accesibilidad del producto final. Asimismo, muchas de estas herramientas nos permiten modificar el material multimedia hasta hacerlo accesible [NCAM, 2006]. De esta forma, se pueden crear contenidos audiovisuales con herramientas software de autor que integren subtitulado y/o audiodescripción, o se pueden editar para incluir subtitulado en material audiovisual pregrabado. De la gran variedad de tecnologías existentes, haciendo una revisión y basándonos en lo relacionado a tecnología accesible para contenidos multimedia nos encontramos con:

- Lenguajes y formatos para sincronizar, fundamentales para conseguir la accesibilidad. Destacamos QuickTime [Apple, 2011 b], SMIL [W3C, 2008 c], SAMI [Microsoft, 2003] o Timed Text [W3C, 2006 a].
- Reproductores como Real Media [Realnetworks, 2008], QuickTime [Apple, 2011 b], Windows Media [Microsoft, 2008], etc.

- Editores de subtitulado y/o audiodescripción para contenidos audiovisuales como MAGpie [NCAM, 2003], Hi-Caption Studio [Hi Software, 2011], etc. o utilidades como Captionmenow de IBM [IBM, 2005], etc.
- Reproductores para contenidos multimedia realizados con Flash accesible de Adobe [Adobe, 2008], [Webaim, 2006] y utilizado por muchos diseñadores.

Todas estas posibilidades a veces no son compatibles unas con otras. Se trata de un mare mágnum de formatos, plataformas, reproductores, lenguajes y tecnologías distintas que hacen del ejercicio de hacer un contenido multimedia accesible algo complicado, pero no imposible. Es importante que se cumplan y sigan los estándares y recomendaciones del W3C, como por ejemplo, que los navegadores y reproductores multimedia cumplan las pautas de accesibilidad para agentes de usuario (UAAG).

Buenas prácticas para elaborar contenidos alternativos

En este apartado se muestran buenas prácticas a la hora de elaborar contenidos alternativos como subtitulados, audiodescripción y lengua de signos.

En cuanto al subtitulado, no hay una normativa específica para su inclusión en Web, pero sí se dispone de documentos que pueden ayudar a su elaboración, como la documentación de buenas prácticas en el subtitulado elaboradas por el CESyA [CESyA, 2006] basadas en la norma AENOR "Subtitulado para personas sordas y personas con discapacidad auditiva. Subtitulado a través del teletexto" [AENOR, 2003]. No obstante, es conveniente mencionar que existen otros documentos que pueden ayudar a complementar la información aquí contenida, entre los que cabe destacar las buenas prácticas para el subtitulado de Joe Clark [Clark, J., 2004].

Sobre la audiodescripción tampoco hay una normativa específica para su inclusión en Web. En España se encuentra reglada su creación por medio de la norma AENOR "Audiodescripción para personas con discapacidad visual. Requisitos para la audiodescripción y elaboración de audioguías" [AENOR, 2005]. Se van a presentar las buenas prácticas de audiodescripción elaboradas por el CESyA [CESyA, 2006] basándose en esta norma.

Buenas prácticas de subtitulado

Dentro del subtitulado existen una serie de aspectos generales que son válidos tanto para programas grabados como en directo.

La adjudicación de colores a los subtítulos dependiendo de los personajes es una estrategia que facilita el seguimiento de la trama argumental y del contexto sonoro, y permite la accesibilidad a personas con discapacidad auditiva.

En el caso de que el color del carácter coincida con el fondo, se puede enmarcar el subtítulo en un recuadro de color para evitar ese problema. Existen ciertas combinaciones de colores que resultan más legibles, porque producen mayor contraste y menos fatiga visual facilitando la lectura, como el carácter amarillo sobre fondo negro, seguido por el carácter verde y el cian sobre el mismo fondo negro. Por tanto, estos son los colores elegidos para aquellos personajes con más presencia en la obra audiovisual.

La norma establece que los subtítulos deben aparecer en la parte inferior de la pantalla ocupando dos líneas y, excepcionalmente, tres. Además dice que para cada personaje se deben asignar líneas distintas. El texto debe estar centrado respecto a la caja.

A la hora de analizar cuándo introducir y sacar un subtítulo, se observan las pausas naturales que hace el ponente, respetando los criterios gramaticales y las unidades lógicas del discurso, o bien los planos. La norma sigue unas guías básicas en la división de subtítulos consistentes en:

- No separar palabras.
- Separar las frases largas según las conjunciones.
- La propia voz muchas veces marca las separaciones mediante pausas o inflexiones.
- Separar las líneas o subtítulos haciéndolos coincidir con comas y puntos.
- Colocar tres puntos suspensivos al final de subtítulo y otros tres al principio del siguiente.

La velocidad recomendada por profesionales del sector experimentados se establece en torno a unos 12 caracteres por segundo.

Además, la comprensión de los subtítulos se mejora con una adecuada sincronización con el sonido. Esto se consigue gracias a la entrada y salida de subtítulos coincidiendo con el movimiento labial, lo que posibilita un apoyo a aquellas personas que poseen restos auditivos. Las voces en off (las emitidas por personajes que no salen en pantalla) van igualmente subtituladas en el color del personaje que la está realizando.

Existen unas pautas particulares en el subtitulado como son:

- No reproducir las abreviaturas.
- Escribir la numeración con letras del cero al diez y con caracteres arábigos el resto de las cantidades.
- Utilizar mejor los paréntesis que los corchetes.
- Utilizar siglas y acrónimos y las formas cortas de entidades u organismos.
- Evitar las muletillas.
- Utilizar los pronombres siempre que se pueda.
- Utilizar las formas cortas de los nombres de personalidades y cargos.

Los subtítulos, en la medida de lo posible, deben ser literales. Además, no hay que olvidar describir el máximo de efectos sonoros necesarios para un buen seguimiento del argumento (a ser posible en la zona superior de la pantalla). También debe aparecer la información contextual, aunque ésta suele ir entre paréntesis y en la misma línea del subtítulo correspondiente [Moreno, L. et al., 2008 b].

Buenas prácticas de audiodescripción

La audiodescripción es un servicio destinado a personas ciegas, tanto totales como con resto de visión, con ceguera congénita o adquirida. Por tanto, completa las necesidades del colectivo de personas con ceguera total, favoreciendo a aquellos con deficiencia visual, además de beneficiar a personas con problemas perceptivos y cognitivos

La norma AENOR UNE 153020:2005 habla de seis requisitos necesarios para audiodescribir:

- Análisis de la obra: no todas las obras audiovisuales permiten una buena audiodescripción. Para saber si una obra puede ser audiodescrita se analizan distintos criterios en un primer visionado, como la existencia de huecos de mensaje para

introducir información, así como la saturación o ausencia de dicha información y que se realice en el mismo idioma de la información sonora de la obra.

- Confección del guión: el guión está formado por unidades de información incluidas en los huecos de mensaje. Para que el guión sea coherente, el audiodescriptor debe consultar la documentación referente al entorno y la temática de la obra. Así, la información se adecuará al tipo de obra y a las necesidades del público a la que se dirige. Además, existen otras características a tener cuenta, como la trama de la acción dramática, los ambientes y los datos plásticos. No hay que olvidar que el estilo de escritura debe ser sencillo, fluido, con frases de construcción directa, terminología específica y adjetivos concretos. La norma UNE 153020 asegura que debe aplicarse la regla espacio-temporal [...] respetarse los datos que aporta la imagen [...] no descubrir ni adelantar la trama y evitar transmitir cualquier punto de vista subjetivo.
- Revisión y corrección del guión: las correcciones necesarias que se hagan servirán para adecuar el guión a las normas de audiodescripción. La norma asegura que deberían ser revisadas las correcciones por una persona distinta del descriptor e incorporarlas después al guión final.
- Locución: se realiza en presencia de la imagen que se describe y el locutor se selecciona según el tipo de voces y el tono adecuado para la obra, de manera que esas voces sean siempre claras para los oyentes. Se debe evitar la entonación afectiva, procurando realizar locuciones neutras.
- Montaje. En la mezcla se equiparan los volúmenes, efectos de ambiente y ecualizaciones con la B.S.O (banda sonora original).
- Revisión: según la norma, una vez finalizada la grabación en el soporte elegido para el caso, debe comprobarse que el producto audiodescrito cumple los requisitos.

3.3.2. Segundo Eslabón: El acceso al contenido multimedia debe ser accesible

Pasamos al segundo eslabón de la cadena; como se ha mencionado con anterioridad, no servirá de nada haber realizado un contenido accesible si el acceso al mismo no goza igualmente de accesibilidad.

Básicamente se pueden distinguir dos formas distintas para ofrecer un contenido multimedia audiovisual en Web a un usuario: mediante descarga del contenido en el equipo del usuario o mediante la emisión del contenido por medio de un reproductor incluido en la página web.

Descarga directa del contenido

Los problemas con los que se encuentran los desarrolladores web para incluir contenidos audiovisuales de manera accesible y que se verán en el apartado 2.3.2 hacen que la opción de descarga directa sea muy utilizada. En este caso, se proporciona al usuario un enlace desde el que poder descargar el contenido, corriendo la reproducción del mismo por parte del software que el usuario tenga instalado en su propio equipo.

A pesar de ser una buena opción para ofrecer multimedia accesible, la descarga directa hay que realizarla teniendo en cuenta que se va a realizar un almacenamiento de información en el equipo del usuario y por lo tanto hay que avisarle claramente de este hecho. Esta situación en las pautas de accesibilidad al contenido web 1.0 (WCAG 1.0) quedaría recogida en la pauta 13.1

que dice: “Identifique claramente el objetivo de cada vínculo” [prioridad 2]. Y en las WCAG 2.0 quedaría recogida en los criterios de éxito 2.4.4 [nivel A] y 2.4.9 [Nivel AAA].

Por otra parte hay que tener en cuenta que las alternativas generadas como son el subtitulado y la audiodescripción pueden venir integradas en el vídeo (“open”) o haber sido generadas como elementos independientes (“closed”) para poderlos activar en función de las necesidades del usuario. En este último caso, será necesario proporcionar todos los elementos en la descarga de forma que el usuario pueda seleccionar en su propio equipo lo que desea reproducir.

Por consiguiente, si se ofrecen contenidos audiovisuales por descarga directa, no sólo habrá que cumplir las pautas vistas en el apartado 2.2, sino que habrá que ser cuidadoso con el texto del enlace con el que se activa la descarga y su etiquetado, de lo contrario no se alcanzaría el nivel AA de adecuación de las WCAG 1.0, ni de las WCAG 2.0.

Emisión del contenido. Técnicas

Pese a que la descarga directa suponga una forma sencilla de proporcionar al usuario el acceso a un contenido audiovisual accesible, la difusión del contenido a través de un reproductor integrándolo en la propia página web es la forma más utilizada.

Ello es debido a que es menos complejo para los usuarios, por ser una opción más usable al acceder a ella por medio de la navegación. Al integrarlo en la Web podemos realizar la emisión del vídeo de dos formas distintas:

- **Emisión en descarga progresiva (falso *streaming*):** con este tipo de emisión el contenido se va descargando en el equipo del usuario y cuando alcanza un porcentaje determinado de descarga se empieza a reproducir. Este tipo de emisión supone una gran ventaja respecto a la descarga ya que no es necesario que el archivo se haya almacenado por completo para iniciar su reproducción. Sin embargo, hasta que el contenido no se ha descargado en su totalidad no se permite hacer uso de los controles de reproducción (reproducir, parar, pausa, etc.) en su totalidad.
- ***Streaming:*** es una descarga del contenido bajo demanda; este tipo de emisión tiene la ventaja de que el usuario puede manejar completamente los controles de reproducción desde un principio y además no es necesario almacenar el vídeo en el equipo del usuario.

Para estas opciones, hay dos maneras de implementación y ambas están asociadas a un reproductor: la primera es activar la descarga con un enlace, forma más frecuente en la opción de *streaming*, y la segunda integrando el contenido audiovisual a través del elemento <object> del XHTML. En cualquiera de los dos casos, hay que cumplir con las WCAG.

Para la opción de activación del enlace, al igual que se ha comentado en la opción anterior de descarga directa, es necesario etiquetar correctamente dicho enlace para cumplir con la pauta 13.1 de las WCAG 1.0, o los criterios de éxito 2.4.4 Y 2.4.9 de las WCA 2.0.

3.3.3. Tercer eslabón: Hay que ofrecer alternativas atendiendo a preferencias del usuario y la interacción del usuario al acceder a contenido debe ser usable

Por último, el acceso del usuario al vídeo debe ser además de posible, intuitivo en la interacción por lo que hay que proporcionar accesibilidad en la forma de mostrar el acceso y control de la información por el usuario. De esta manera el punto de verificación 14.1 (*Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.* [Prioridad 1]) se debe cumplir de acuerdo a la WCAG 1.0.

Como técnicas para el punto 14.1 recomendamos seguir las técnicas de Diseño Centrado en Usuario [Henry S., 2007], donde el usuario participe en el diseño y en la elaboración de los contenidos así como la aplicación de Reglas de Lectura Fácil; estas técnicas se identificarán como Técnicas del Lenguaje en la tabla resumen mostrada en el siguiente apartado.

Además, los usuarios deberían poder tener acceso a la reproducción del vídeo, de acuerdo a sus características y preferencias. Hay factores que deben tenerse en cuenta, tales como el tamaño del vídeo, la duración del vídeo, la información de progreso en la reproducción, la velocidad y el tipo de conexión del usuario, el agente de usuario asociado a la reproducción, el formato del vídeo, si el usuario va a tener el control, etc. No todos estos factores son siempre determinantes, dependerá de la modalidad que haya sido elegida al servir el contenido.

Por ejemplo, la información del tamaño del vídeo será importante para el usuario en las opciones de descarga o descarga progresiva, pero no es necesario en streaming.

En función de estos tipos, las posibles características a considerar son las mostradas en la tabla 9.

Características	**Descarga directa**	**Descarga progresiva**	**Streaming**
Tamaño del recurso	SI	SI	NO
Duración del contenido audiovisual	SI	SI	SI
Opciones de velocidad y tipo de conexión para preferencias del usuario	SI*	SI*	SI*
Barra de progreso	NO	SI	NO
Agente/s de usuario asociados a la reproducción	SI	SI	SI
Formatos (Windows media, Real, QuickTime, SMIL)	SI	SI	SI
Se permite control al usuario	NO*	NO*	SI
***A excepción de SMIL**			

Tabla 9: Información a presentar al usuario

Esta situación queda recogida en el punto de verificación 11.3 (Proporcione la información de modo que los usuarios puedan recibir los documentos según sus preferencias (Por ejemplo, idioma, tipo de contenido [Prioridad 3]) de WCAG 1.0.

Aparte de ofrecer toda la información al usuario comentada anteriormente, es importante considerar cómo hay que presentar esa información al usuario. Como Técnicas para el punto 11.3 está la aplicación de los principios de Diseño Universal [Shneiderman, B, 2000] y criterios de usabilidad en el interfaz web donde se incluya el contenido multimedia y sus características de acceso. Denominaremos a dichas técnicas" Técnicas de Diseño Universal", para identificarlas en la tabla resumen ofrecido en el siguiente apartado. Puede ser muy conveniente contar en el diseño y desarrollo con una revisión experta humana de usabilidad, además de una validación por usuarios para garantizar que el usuario puede acceder con facilidad al contenido.

3.3.4. Tabla resumen (WCAG 1.0|WCAG 2.0) de la accesibilidad a los contenidos audiovisuales en la web

Como soporte para diseñadores y desarrolladores que quieran incluir contenidos audiovisuales en la Web, considerando la accesibilidad de una manera completa, teniendo en cuenta los tres eslabones de la cadena vistos, se presenta a continuación en la tabla 10 una tabla resumen, que incluye por cada eslabón, los puntos de verificación de las WCAG 1.0 y los criterios de éxito de las WCAG 2.0 relacionados junto con sus técnicas como ayuda.

La cadena de la accesibilidad de los contenidos audiovisuales en la Web **Listado de eslabones**		**WCAG 1.0** **Puntos de verificación**	**WCAG 1.0** **Prioridad**	**WCAG 2.0** **Criterios de Éxito**	**WCAG 2.0** **Nivel de Adecuación**	**Técnicas WCAG 2.0 ,SMIL y otras**
(1) El contenido en sí mismo debe ser accesible		**1.1**	**1**	**1.1.1 Non-text Content**	**A**	**G68**
				1.2.1 Audio-only y video-only (Prerecordes)	**A**	**G158, G159**
				1.2.9 Live Audio-only	**AAA**	**G150, G157**
		1.3	**1**	**1.2.5 Audio Description**	**AA**	**G78, SM6, SM7**
				1.2.7 Audio Description (ext.)	**AAA**	**G8, SM1,SM2**
		1.4	**1**	**1.2.2 Captions (Prerecorded)**	**A**	**G93, G87, SM1, SM2**
				1.2.3 Audio Description or FullText Alternative	**A**	**G69, G78**
				1.2.4 Captions (Live)	**AA**	**G9, G93**
				1.2.8 Alternative Full Text	**AAA**	**G69, G159**
		No hay correspondencia		**1.2.6 Sign Language**	**AAA**	**G54, G81, SM13, SM14**
(2) El acceso al contenido debe ser accesible	**Descarga completa** **Descarga progresiva *streaming***	**13.1**	**2**	**2.4.4 Link Purpose (In Context)**	**A**	**G91, G53**
				2.4.9 Link Purpose (Link Only)	**AAA**	**G91, C7**
	Descarga progresiva *streaming*	**6.3**	**1**	**No hay correspondencia directa**	**A**	**H46, H53,Técnicas objeto**
		3.2	**2**	**4.1.1 Parsing**	**A**	**G134, G154**
		No hay correspondencia		**4.1.2 Name, Role, Value**	**A**	**G10, G108, 135**
(3) Hay que ofrecer alternativas atendiendo a las preferencias del usuario; además, la interacción del usuario al acceder al contenido debe ser usable		**14.1**	**1**	**No hay correspondencia directa pero considerar:3.1.5 Reading Level**	**AAA**	**G86, G103, G79Técnicas Lenguaje**
		11.3	**3**	**No hay correspondencia directa pero considerar:2.2.2 Pause, Stop, Hide**	**A**	**G4, G11Técnicas Diseño Universal**

Tabla 10: Tabla resumen para diseñadores sobre cómo ofrecer accesibilidad a los contenidos audiovisuales en la Web [Moreno L. et al., 2008 a]

Capítulo 4

4. Accesibilidad en elementos relativos al contenido multimedia en HTML5

En este capítulo se va a explicar qué elementos de HTML5 están relacionados con la accesibilidad y el contenido multimedia tan extendido actualmente. Desde el video, pasando por el soporte de subtítulos, requisito de accesibilidad en los contenidos multimedia tal como se ha visto en el capítulo anterior. Como se mencionará más adelante no hay soporte de subtitulado hasta el momento para incluirlo directamente con HTML5, pero sí existen soluciones a integrar de otras tecnologías tal como se ha utilizado en este proyecto, y que se verán en el capítulo cinco.

Se hablará del elemento <video> y su independencia de usar otros lenguajes para reproducir un archivo de vídeo en formatos de sobra conocidos actualmente, y al igual con el elemento <audio> que facilitan muchísimo las cosas para introducir contenido multimedia. Éstas son unas de las más importantes, pero no hay que despreciar al resto, ya que se han mejorado las etiquetas dándoles un mayor significado, más intuitivas, sobre todo mejorando los formularios y los campos de los que están formados, ayudando a los motores de búsqueda.

4.1. Elemento <video>

Hasta este momento como se ha explicado en apartado 2.6.1, el método para incluir vídeo en un sitio web ha requerido un plug-in de terceros, dependiendo la calidad, la interfaz de usuario y la accesibilidad, del plug-in de vídeo utilizado. Con el HTML5 no es necesario al incorporar el elemento con la etiqueta estándar <video>.

Respecto a la accesibilidad de <video>**,** y a la espera de las especificaciones de la misma y las implementaciones de los distintos navegadores, es posible que resuelva dos de los mayores problemas que se presentan en la actualidad: la accesibilidad del teclado y el soporte de subtítulos**.**

4.1.1. Control del teclado

De forma predeterminada, la etiqueta <video> de HTML5 utiliza los controles de reproducción que proporciona el navegador, que debe permitir la navegación con el teclado. Aunque es todavía demasiado pronto para saber si todos los fabricantes de navegadores incorporarán esta funcionalidad correctamente, es probable que sea una gran mejora con respecto lo que habitualmente vemos hoy.

4.1.2. Subtítulos

La etiqueta <video> en HTML5 no contiene ningún mecanismo que de soporte para incluir el subtitulado, la audiodescripción, transcripción, u otras alternativas multimedia de manera sincronizada.

En otras palabras, los desarrolladores que quieran incluir vídeo con subtítulos no disponen de elementos HTML5 que les permiten incluirlo directamente. Ahora bien, esto no es peor que la situación que tenemos hoy, y, parece ser que, en los grupos de Accesibilidad del W3C están buscando alguna forma de solventar el tema con el nuevo estándar.

De momento la solución más factible y directa es la integración de otras tecnologías como es JavaScript o Flash para crear un efecto de subtítulos proyectado encima del marco del vídeo, y que a su vez permita mediante controles la opción de mostrar o no los subtítulos. En este proyecto se ha seleccionado la tecnología JavaScript, y no Flash. La justificación de esta elección es la comodidad de no tener que usar plug-ins externos para acceder al subtitulado, evitando el riesgo de que el usuario no pueda visualizarlo correctamente al no tener instalado en su navegador los plug-ins necesarios para usar la tecnología Flash. Así, se asegura que la información sea accesible al mayor número de personas. Esta solución que se propone se muestra en el capítulo cinco.

4.2. Elementos semánticos

HTML5 introduce varios elementos semánticos que representan secciones lógicas o componentes de una aplicación web o un documento: <section>, <nav>, <article>, <aside>, <hgroup>, <header>, <footer>, así como nuevas reglas para el uso de las etiquetas de cabecera <h1>-<h6> y los elementos <dirección>. Las nuevas etiquetas proporcionan una forma más sencilla para los desarrolladores a la hora de definir las distintas partes de un documento.

En general, todo esto tiene un importante potencial para mejorar la accesibilidad, sobre todo si los lectores de pantalla y otros productos de apoyo comienzan a utilizar la información asociada.

4.2.1. Uso exclusivo de <h1> en <section> y <article>

En las versiones actuales de HTML, la única forma de definir las secciones y la jerarquía del esquema de un documento es el uso de la <h1>-<h6> en una estructura plana. Esto es muy beneficioso para la accesibilidad, pero puede ser un poco incómodo para los desarrolladores de páginas web. HTML5 introduce dos nuevos elementos, <section> y <article>, que definen secciones lógicas y artículos para organizar el contenido y los niveles de jerarquía de lo mismo.

Esta solución puede originar algunos problemas de accesibilidad durante la fase de transición hacia el estándar, ya que es posible que se sigan usando las mismas técnicas anteriores de HTML4.01 en HTML5 siendo esto innecesario, per haciendo posible que continuemos usando etiquetas de encabezados aunque estén anidadas dentro de bloques <section> y <article>.

4.2.2. El elemento <nav>

El elemento <nav> en HTML5 proporciona una forma de especificar de forma explícita los elementos de navegación en las páginas. Al igual que antes, esto será realmente destacable cuando los productos de apoyo (y los navegadores web estándar) ofrezcan e a los usuarios la capacidad de tomar ventaja del estándar directamente.

4.2.3. Nuevos tipos para la etiqueta <input> y sus extensiones

HTML5 define 13 nuevos valores para el elemento <input> (búsqueda, teléfono, url, correo electrónico, fecha y hora, fecha, mes, semana, tiempo, fecha y hora local, número, intervalo y color).

Si bien queda por ver cómo cada uno de los nuevos tipos de entrada se representará o será utilizada por los navegadores web. Estos son campos de texto utilizados por secuencias de comandos y se enviarán por GET y POST como tal. En algunos casos (como la fecha y la hora) HTML5 dicta que los datos se almacenarán internamente o serán representados de manera específica, de forma que los cambios en su manera de ser utilizados será mínimos.

Por defecto, para un elemento <input> en HTML se le asigna un type = “text”, así que éste es un cambio perfectamente compatible con versiones anteriores, y no debería causar ningún problema (de accesibilidad o de otro tipo).

4.2.4. Tipo de entrada Search

El tipo de entrada de búsqueda representa un campo de texto que funciona como una caja de búsqueda. Respecto a este nuevo tipo su utilidad y los beneficios para la accesibilidad son bastante grandes, ya que permitiría a las tecnologías de asistencia proporcionar, mediante un solo comando o pulsaciones de teclas, acceder a la casilla de búsqueda de la página actual, independientemente de dónde se encuentre en el contenido.

4.2.5. Tipos de entrada tel, url y email

Mediante estos nuevos tipos, el desarrollador tiene una ayuda para la validación de entrada de los datos, de forma que se puede exigir que el texto introducido cumpla las especificaciones del formato URL o de direcciones de correo. Además, existe la posibilidad de integración con otras fuentes de datos, mediante lo cual, por ejemplo, los agentes de usuario podrían proporcionar un mecanismo para obtener direcciones de email o números de teléfono de un navegador o de la libreta de direcciones y cumplimentar de esta forma los campos de entrada.

La accesibilidad de estas interacciones basadas en el navegador será responsabilidad de los proveedores de navegadores**.**

4.2.6. Tipos de entrada date y time

Seis de los nuevos tipos de entrada están relacionados con fechas y horas en una u otra forma: datetime, date, month, week, time y datetime-local**.**

Es de esperar que los navegadores web y otras aplicaciones de usuario presentará estos campos de entrada como, quizá, calendarios o relojes implementados con pop-ups para permitir una fácil lectura y una selección sencilla de los valores apropiados. Esto se puede esperar que genere un significativo aumento de la accesibilidad, ya que estos reproductores suministrados por los navegadores es probable que sean muy accesibles, mientras que los códigos actuales, basados habitualmente en JavaScript, suelen tener deficiencias en cuanto a accesibilidad. Además, el uso de una interfaz única para la selección de fechas y horas, reducirá la curva de aprendizaje para los usuarios ya que tendrá un formato similar en todos los sitios web**.**

Durante el período de transición, cuando algunos usuarios cuenten con navegadores con soporten HTML5 y otros no, los sitios que actualmente utilizan códigos desarrollados tendrán que buscar alguna forma de determinar las capacidades del navegador y solo hacer uso de estas etiquetas HTML5 si el browser las soporta. Actualmente sólo las versiones más recientes de Opera soportan oficialmente los tipos de entrada date y time, aunque se espera que otros navegadores las soporten en breve.

4.2.7. Tipos de entrada numéricos

Los tipos de entrada number y range se utilizarán para la entrada de valores numéricos, y, lo más probable, es que sea presentado por los navegadores como spinboxes y controles deslizantes, respectivamente. Dado que estos reproductores se corresponden directamente con funcionalidades nativas de la plataforma y estas suelen incluir funciones de accesibilidad, esta se supone que será elevada. La conversión de los campos de entrada de texto para estos campos numéricos no debe crear un nuevo problema de accesibilidad.

4.2.8. Extensiones para la etiqueta <input>

Además de los nuevos tipos <input>, HTML5 también añade cuatro nuevos atributos a todos los elementos de entrada que tienen un impacto potencial de la accesibilidad: autofocus, placeholder, required y pattern. Todos ellos implementan características y funciones que se han estado utilizando en aplicaciones web desde hace años a través de secuencias de comandos, pero ahora el estándar nos permite ser codificadas directamente en HTML.

Los navegadores, exceptuando Internet Explorer, admiten estas extensiones en mayor o menor medida, pero se espera que las implementen en todas sus funcionalidades en breve. Mientras tanto, las soluciones basadas en scripts se puede utilizar junto con los atributos de HTML5, con la incorporación de algunas utilidades para determinar el user-agent del navegador del usuario.

4.2.9. Enfoque automático (autofocus)

El atributo de enfoque automático permite a los desarrolladores especificar que un elemento, de forma particular, debe recibir el foco de entrada tan pronto como la página que lo contiene se carga, de modo que puede empezar a escribir de inmediato sin tener que hacer click específicamente en ese control. Las capacidades de autofoco basado en scripts son muy utilizadas en la actualidad, pero conllevan una serie de problemas, incluyendo el hecho de que no hay forma de deshabilitarlas. Con el atributo autofocus los navegadores web y otras aplicaciones de usuario puede permitir la capacidad de deshabilitar esta característica, ya sea en un sitio web específico o para toda la Web.

Hay que tener en cuenta que existen algunos problemas de acceso con el enfoque automático, y esto no cambiará con el nuevo estándar, pero en los casos los que queramos utilizar esta característica, el atributo de enfoque automático en HTML5 proporciona una manera más fácil y más simple realizarlo.

4.2.10. Marcador de posición de texto (placeholder)

El marcador de posición de texto en los campos de entrada en HTML está considerado como un requisito de accesibilidad. Actualmente se implementa con un texto por defecto que contiene la entrada o el área de texto de los formularios, siendo, este texto, eliminado automáticamente por un script cuando se incluye texto por parte del usuario.

El atributo placeholder de HTML5 permite a los desarrolladores web especificar el marcador de posición del texto directamente en el código del formulario. Esto ahorra el desarrollador el problema de tener que utilizar una solución basada en scripts, y permite a los proveedores de navegadores incluir tecnologías para ayuda de una manera consistente. Además, el uso generalizado del atributo (en lugar de secuencias de comandos existentes) nos asegura que el texto del marcador de posición tendrá un aspecto y comportamiento, a lo largo de la Web, uniforme, incrementando la accesibilidad y facilitando el aprendizaje de uso de los sitios web.

Hay que tener en cuenta que está explícitamente prohibido por la especificación HTML5 dejar sin etiquetas explicitas los marcadores de posición de texto.

4.2.11. Campos requeridos (required)

La capacidad de indicar que un campo es requerido ha sido posible con ARIA desde hace algún tiempo, e incluso se apoyó en los navegadores más modernos y los lectores de pantalla. Esta solución es mucho mejor que la comúnmente utilizada de “añadir un asterisco al texto de la etiqueta” ya que permite a los agentes de usuario y tecnologías de apoyo informar al usuario de que un campo en particular se requiere. HTML5 va un paso más allá añadiendo una característica directamente a la sintaxis HTML (no requiere ARIA), con la idea de que el navegador no sólo mostrará los elementos requeridos de forma diferente, sino que proporcionará automáticamente los mensajes de error (en una forma accesible, suponemos) cuando alguien intenta enviar un formulario sin haber llenado en todos los valores requeridos.

4.2.12. Patrones (pattern)

El atributo pattern de HTML5 incrementa las capacidades del atributo required al indicar que un campo de entrada particular debe tener un valor que coincide con una cierta

expresión regular definida por el desarrollador. Esto permite recibir los datos estrictamente con un formato predefinido (por ejemplo, en el caso de los números de tarjetas de crédito) de forma que, la validación automática del lado del cliente, permite reportar errores.

La especificación exige que, cuando se utiliza el atributo patrón, también se añada un título que explique los detalles del modelo que se verificará.

Si este atributo no nos cubre todas las necesidades de nuestra aplicación, HTML5 proporciona una interfaz para acceder a las funciones de validación de formularios a través de secuencias de comandos.

4.3. Elemento Track

Este nuevo elemento todavía en desarrollo, está orientado a otras etiquetas de HTML5 como <video> y <audio>, el cual está relacionado con el tiempo de reproducción de estas etiquetas. Por sí misma no represente nada, pero si se usa junto a etiquetas multimedia como las mencionadas anteriormente, aporta una gran accesibilidad ya que permite relacionar texto directamente basándose en el tiempo de estas misma etiquetas.

Como se ha mencionado anteriormente en el apartado 2.6.4, todavía está en desarrollo, pero es posible que esta etiqueta sea muy importante de cara a la accesibilidad, puesto que ahorrará tiempo y esfuerzo para hacer más accesibles elementos multimedia

Capítulo 5

5. Diseño y desarrollo de interfaz basado en HTML5

En este capítulo se va a proceder a la explicación y descripción del diseño e implementación de una página web[8] en HTML5. Esta página contiene una interfaz para reproducir contenido multimedia vídeo accesible, de acuerdo a todo lo comentado en el capítulo cuatro.

Para el diseño de la interfaz, se ha tenido muy en cuenta todas las normas y estándares relacionados con la accesibilidad en contenido multimedia, concretamente las WCAG y UAAG. Los principales objetivos de este proyecto satisfacen estas normas, que son la inclusión de alternativas al acceso a la información multimedia como el subtitulado para sordos, pudiendo elegirlos en dos idiomas (español e inglés, y pudiéndose ampliables) y la audiodescripción.

Una vez satisfechos los principales requisitos, se llevó a cabo una implementación que posibilitara el manejo de todos los controles de la interfaz por teclado, además de por ratón, y se introdujeron símbolos convencionales de las distintas funcionalidades de un reproductor multimedia para que fueran fácilmente reconocibles por los usuarios.

El primer paso fue hacer el diseño de la presentación abstracta, es decir definir la estructura lógica de la página web. La implementación fue sencilla gracias a las nuevas etiquetas comentadas en el apartado 3.4. HTML5 resulta muy intuitivo ya que estas etiquetas tienen nombres representativos para lo que están destinadas. Para ello se estudiaron las nuevas etiquetas principales, y se hizo una página con una estructura muy similar a las que se pueden observar en la figura 5 en el apartado 3.4.1., en el que se seleccionaron las etiquetas más importantes para dar forma al HTML, que son <header>, <nav>, <section>, <article>, <aside> y <footer>.

A continuación se hizo el diseño de la presentación concreta, y su implementación con CSS. Así, se ha obtenido el aspecto gráfico deseado mediante las técnicas habituales de estilo,

[8] Disponible en: http://labda.inf.uc3m.es/LourdesPlayer/

ya que el efecto de CSS es el mismo en HTML5 que en HTML4, con la mínima diferencia que ahora como las etiquetas principales tienen nombres muy representativos, se pueden dar estilos más cómodamente que antes.

Una vez implementado la interfaz en HTML5 y CSS, se paso a implementar las funciones necesarias para la reproducción de video. Cómo tecnologías ha sido necesario incorporar tecnología JavaScript para la incorporación de diversos controles necesarios según requisitos de diseño tal como se muestra en el apartado 5.1.2. La explicación de la elección de la tecnología JavaScript y no de otras como puede ser Flash se hará en el apartado 5.3.

5.1. Requisitos de accesibilidad de un contenido multimedia en la Web

Como ya se vio en el apartado 3.2.1, hay que ofrecer accesibilidad al contenido audiovisual en la Web siguiendo recomendaciones del estándar WCAG y UAAG. Para proveer de accesibilidad, se debe de dar soporte de acceso al contenido, y hay que tener en cuenta distintas dimensiones en el acceso como son:

1. Contenido en sí: el contenido debe ser accesible.
    - Esto significa conseguir que el contenido en sí mismo sea accesible, lo que implica proveer de alternativas sincronizadas como el subtitulado y la audiodescripción, entre otros, siguiendo las WCAG.
2. Cómo llegar al contenido: el acceso a ese contenido deber ser accesible.
    - Básicamente hay dos formas distintas de ofrecer un contenido multimedia audiovisual en Web a un usuario: mediante descarga del contenido en el equipo del usuario o mediante la emisión del contenido por medio de un reproductor incluido en la página web.
3. Visualización del contenido: hay que ofrecer alternativas atendiendo a preferencias del usuario, y la interacción del usuario al contenido debe ser usable, siguiendo las UAAG.
    - El acceso del usuario al vídeo debe ser además de posible, intuitivo en la interacción por lo que hay que proporcionar accesibilidad en la forma de mostrar el acceso y control de la información por el usuario, tales como barra de progreso, control al usuario en la reproducción, duración del contenido audiovisual, tamaño del recurso, agentes de usuario asociados a la reproducción, tal y como se explica en el apartado 5.2 y se dará una explicación más detallada de la implementación a nivel de código en el apartado 5.3.

5.2. Diseño de Interfaz que reproduzca contenido multimedia accesible

En este apartado se explicará el diseño de la interfaz que hace que el vídeo se pueda reproducir de manera accesible. Tal y como se ha indicado, las UAAG muestran cómo hacer que los agentes de usuario sean accesibles para personas con discapacidad, más concretamente dan soporte a que el contenido web sea accesible siguiendo las WCAG. Para el diseño de esta interfaz se han seguido algunas de estas pautas y por consiguiente se han tenido en cuenta siguientes requisitos:

- **Requisito 1:** Que todo el contenido sea accesible tanto por teclado como por ratón.

Para la realización de este requisito se ha implementado un script que permite al usuario manejar la interfaz del vídeo tanto como por ratón como por teclado. Para hacer efectivo el control por teclado a través de un script de JavaScript, es necesario que el usuario haya usado el botón izquierdo del ratón en cualquiera de los elementos que forman el video, ya sea el vídeo en sí para reproducir el clip de vídeo o los controles situados al pie del video, o por el contrario, se mueva previamente por cualquiera de los botones de la barra de control mediante la tecla tabulación. Una vez realizada cualquiera de estas dos acciones, el usuario podrá controlar el vídeo a través del teclado.

Para más información se podrá seleccionar el botón de ayuda situado en la barra de controles, en el que se mostrará un recuadro con información sobre las teclas que debe pulsar para el completo control del contenido multimedia.

- **Requisito 2**: Que el usuario seleccione la forma en que quiere que se muestre el contenido.

El usuario podrá seleccionar la forma en que quiera acceder al contenido. Para ello se ha dotado al contenido multimedia de accesos alternativos a la información, como son subtítulos en varios idiomas (español e inglés), estando por defecto la opción en español y ampliable a varios idiomas, la opción de escuchar la audiodescripción del vídeo aislado o combinado con el audio del video, y escuchar también el audio del vídeo aislado de la audiodescripción, todas ellas con la posibilidad de activar o desactivar dicha opción mediante un botón en la barra de controles asociado a cada uno de las opciones anteriormente comentadas.

- **Requisito 3**: Que el usuario tenga el control sobre la interfaz del usuario, es decir, que la interfaz sea lo suficiente accesible y fácil de entender, como para que el usuario pueda controlarlo sin dificultad.

La interfaz de contenido multimedia se ha diseñado de forma que sea lo más accesible posible, dotando a los controles de un significado simbólico muy reconocible con el objetivo que el usuario tenga el control sobre dicha interfaz. Para ello se han agrupado los botones relacionados directamente con el control del vídeo a la izquierda, con símbolos reconocibles por cualquier persona familiarizado con la reproducción de vídeo en cualquier formato como reproducir/pause, parar, retroceder y avanzar, seguido del control para el volumen del vídeo con un botón para silenciar el volumen o bajar y subirlo a elección del usuario.

Para cumplir con los requisitos definidos, hay una serie de controles necesarios a incluir en la interfaz tales como habilitar/deshabilitar subtítulos, habilitar/deshabilitar audiodescripción, habilitar/deshabilitar el audio del propio video, mostrar información de las teclas de acceso rápido por teclado, y elegir el idioma del subtítulo mediante dos botones característicos de dichos idiomas.

Para diseñar la disposición espacial de todos estos controles, se diseñó una maqueta gráfica que se muestra en la figura 21:

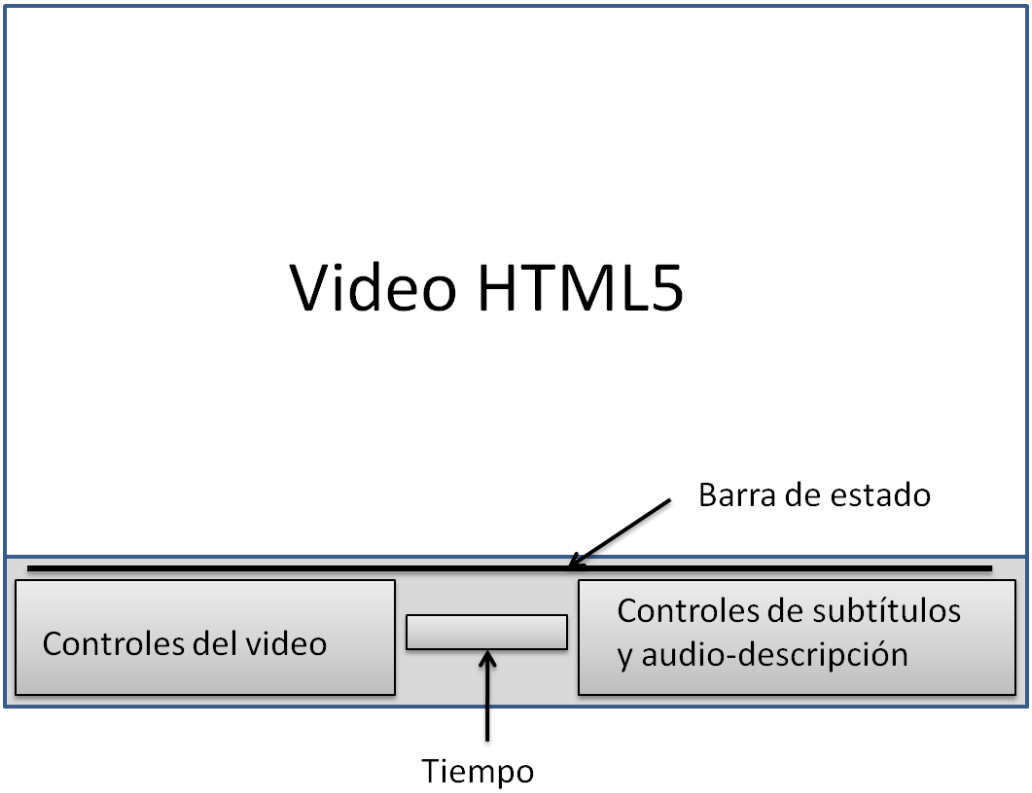

Figura 21: Boceto de la interfaz en HTML5

Se diseñó una barra de estado donde se incluían de izquierda a derecha los siguientes elementos: los controles nativos de un reproductor, luego una barra de progreso del tiempo, y después los controles orientados a controlar los recursos de accesibilidad como el subtitulado y la audiodescripción.

Aun definiendo controles en base con los requisitos definidos que responden a las pautas UAAG 2.0, el reproductor no cumple con la totalidad de las pautas tal como se documenta en el anexo de esta memoria, obteniendo un nivel de conformidad A de las UAAG 2.0.

5.3. Implementación en HTML5 de la Interfaz.

En este apartado se va a proceder a describir por un lado la estructura de la página, el contenido a incluir y por otro lado, la guía de implementación de la página web creada en HTML5 en la que se ha incluido el reproductor accesible.

Actualmente, HTML5 no aporta unos controles nativos para la etiqueta <video> que den soporte a los requisitos de accesibilidad definidos para la interfaz. Los controles nativos del HTML5 son demasiado básicos y no permiten al usuario tener un control completo sobre el video, además de que aunque el W3C anunció que se está trabajando en incluir subtitulado en el nuevo estándar, todavía no está implementado. Esta ha sido la causa de integrar con HTML5 otras tecnologías como es el caso de JavaScript. Como opciones de tecnologías a integrar para poder implementar una interfaz que cumpla con los requisitos definidos, estaban de opciones además de JavaScript la tecnología Flash, se optó por la primera, ya que no necesitaba de plug-ins externos para su uso.

5.3.1. Controles del reproductor

A continuación se enumeran las funcionalidades de los controles nativos de HTML5 para la etiqueta <video>:

- Botón de reproducir/pause.
- Barra de progreso e información del tiempo incluido en él y posibilidad de seleccionar el minuto de visualización.
- Botón para habilitar pantalla completa.
- Controles de volumen.

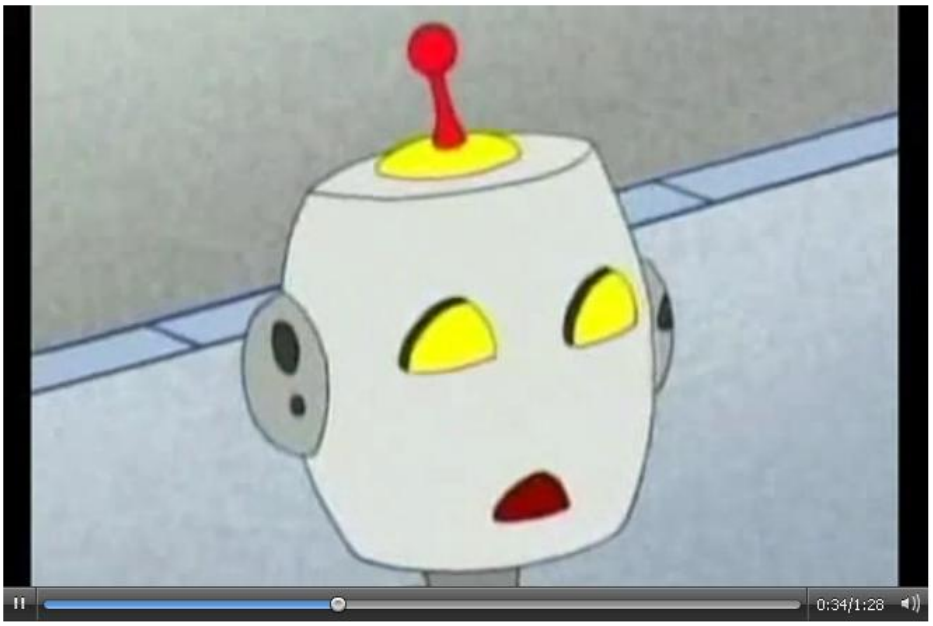

Figura 22: Controles nativos incluidos en HTML5

Para cumplir con los requisitos, estos controles son insuficientes ya que se necesita un control más extenso del vídeo y también la posibilidad de mostrar subtítulos y audiodescripción, y control del vídeo por teclado de acuerdo a los requisitos definidos. Las carencias principales son:

- HTML5 no aporta, de momento, la introducción de subtítulos de forma directa en código.
- No es posible introducir audiodescripción al igual que los subtítulos.
- Los controles directos del vídeo son demasiado simples, privando al usuario de un control más completo sobre dicho video.
- No permite controlar el vídeo a través del teclado.

Estas carencias se pretenden solucionar con la incorporación de unos controles creados en JavaScript que aporten subtitulado en varios idiomas y audiodescripción, controles específicos para cada elemento y otros necesarios para cumplir con los requisitos de la interfaz definidos, que son los siguientes:

- Botón de reproducir/pausa.
- Botón de parar.
- Botón para retroceder el vídeo 5 segundos.
- Botón para avanzar el vídeo 5 segundos.

- Botón para silenciar el video.
- Barra de control y botones para aumentar y disminuir el volumen un 10%..
- Botón para habilitar/deshabilitar subtítulos.
- Botón para habilitar/deshabilitar audiodescripción.
- Botón para habilitar/deshabilitar audio del video.
- Botón de información sobre las teclas rápidas de control por teclado.
- Control de selección de idioma de los subtítulos.
- Barra de progreso del vídeo con opción de seleccionar el minuto de reproducción.

El uso de este lenguaje JavaScript complementario a HTML5 es simple. Su sintaxis no es complicada ya que es un lenguaje basado en objetos, y con el que es fácil de manipular tanto contenidos multimedia, como en este caso videos, como animaciones básicas en un documento HTML aportando un aspecto visual dinámico y atractivo, con la ventaja de que solo es necesario incluir en la cabecera del documento HTML el enlace del fichero JavaScript que se quiere usar.

En el caso de que JavaScript estuviese desactivado en el navegador, se podrá acceder al video mediante los controles nativos de HTML5, pero sin la posibilidad de utilizar los controles anteriormente descritos ni acceder a los subtítulos y audiodescripción. Este aspecto es importante ya que aporta accesibilidad si se da el caso de que el usuario no tenga activado JavaScript.

5.3.2. Elementos de HTML5 utilizados

Para la realización de esta página se han usado muchos elementos nuevos introducidos en HTML5, pero también bastantes tomados de la anterior versión HTML4.01, ya que hay numerosos elementos que no se han eliminado y hasta ahora son útiles para ciertos casos.

Etiquetas nuevas usadas:

- <header>: articulo introductorio para artículos o navegación, utilizados tanto para la página web, como para cualquier esquema diseñado como introducción a algo.
- <nav>: diseñado para crear barras de navegación muy comunes en páginas web, donde normalmente figuran las distintas secciones o menús de un sitio web.
- <section>: representa una sección genérica de un documento o una aplicación. Podría agrupar contenido de un tema específico dentro de un artículo en que se pueda introducir.
- <video>: nueva etiqueta que permite introducir un vídeo dentro de una página web sin necesidad de software adicional.
- <audio>: nueva etiqueta que permite introducir una pista de audio dentro de una página web sin necesidad de software adicional.
- <button>: etiqueta ideada para introducir un botón con el fin de crear una acción o evento.
- <source>: se usa principalmente en las etiquetas <video> y <audio> para indicar la dirección donde se encuentra el archivo con el que se quiera interactuar.
- <aside>: división lógica de un apartado independiente dentro de la página relacionado con el contenido que lo rodea. Se puede usar como una sección vertical de información adicional al contenido de la página.
- <footer>: representa el pié de una sección, con información acerca de la página/sección que poco tiene que ver con el contenido de la página.

- <article>: representa un componente de la página o sitio web, con la intención de que pueda ser reutilizado y repetido.
- <form>: sección especialmente creada para introducir etiquetas de formularios.

Etiquetas usadas ya permitidas en HTML4.01:

- <head>: sección inicial donde se introduce el título de la página y los archivos necesarios de CSS y JavaScript.
- <body>: sección principal de un archivo HTML donde se introduce el grueso de la información que se necesita para la construcción de una página web.
- <title>: en la etiqueta <head> indica el nombre de la página web, y si se introduce en otras etiquetas como atributo, sirve para mostrar un pequeño texto encima del puntero del ratón.
- <link>: etiqueta para enlazar archivos CSS en la página web.
- <script>: etiqueta para enlazar archivos JavaScript en la página web.
- <h1>,<h2>,<h3>: etiqueta para mostrar, normalmente, cabeceras de texto en cualquier sección de la página web.
- <p>: define principalmente un párrafo de texto o elementos en html.
- : etiqueta para dividir secciones lógicas dentro de una página web.
- : define una sección cualquiera de un documento.
- : etiqueta el inicio de una lista sin orden.
- : define uno de los elementos de una lista sin orden.
- <a>: normalmente se utiliza para señalar un hipervínculo.
- <label>: define un campo para formularios.
- <input>: define una entrada normalmente para formularios.

5.3.3. Estructura lógica de la página creada en HTML5

A continuación se explicará detalladamente cada sección de la que está formada la página creada para este proyecto, en la que se aplican los conocimientos que se han obtenido sobre HTML5 y sus elementos y etiquetas a lo largo del documento. Finalmente, se explicará el método seguido para dotar al vídeo de subtítulos y audiodescripción.

Cabecera

En esta sección introduciremos los enlaces relacionados con fichero CSS y JavaScript necesarios para el correcto funcionamiento de la página, que se encargará de dar formato gráfico y estructura a los elementos del fichero en HTML5, y responsable de que el contenido multimedia incluido en este proyecto se visualice correctamente, respectivamente.

Claramente se aprecia que esto es idéntico a anteriores versiones de HTML5 y que es imprescindible para su correcta visualización. El código responsable de lo comentado anteriormente se puede ver en la figura 23.

```
<head id="head" >
      <title>Proyecto fin de carrera</title>
      <link rel="stylesheet" href="prueba1.css"
type="text/css" media="screen" />
      <link rel="stylesheet" href="css_barra.css"
type="text/css" media="screen" />
      <script type="text/javascript" src="java.js"></script>
      <script type="text/javascript"
src="progressbar.js"></script>
      <script type="text/javascript"
src="cargarXML.js"></script>
</head>
```

Figura 23: Cabecera de la página HTML5

Navegación y título

La sección navegación contendrá una barra de navegación horizontal en la que se mostrará un menú con los que se podrá dirigir a las distintas secciones de la página, y el título para mostrar con una frase el nombre de la página.

Para ello, simplemente se usarán dos etiquetas nuevas de HTML5, <header> para mostrar el título de la página mediante una etiqueta de encabezado <h1>, y <nav> para mostrar el submenú inicial de la página mediante las etiquetas de listas sin orden ya presentes en anteriores versiones de HTML, y la cual se muestra en la figura 24.

```
<header>
<h3>Result of final year project</h3>
</header>
<nav>
      <ul>
            <li class="Blog"><a href="#">Blog</a></li>
            <li><a href="#principalblog">Information</a></li>
            <li><a href="#titulo_player">Video</a></li>
            <li><a href="#fr">Further reading</a></li>
            <li><a href="#mul">Multimedia</a></li>
            <li><a href="#pie">Contact</a></li>
      </ul>
</nav>
```

Figura 24: Código HTML5 de navegación e introducción

Con esto conseguiríamos una lista de elementos independientes que posteriormente daremos formato y estructura para reordenarlo horizontalmente, aplicando estilos a través de CSS como se puede observar en la figura 25.

```
nav {
position: absolute;
left: 0;
width: 100%;
background: black;
margin-bottom:50px;
}

nav ul {
margin: 0 auto;
width: 940px;
list-style: none;
}

nav ul li {
float: left;
}

nav ul li a {
display: block;
margin-right: 20px;
width: 140px;
font-size: 14px;
line-height: 44px;
text-align: center;
text-decoration: none;
color: #777;
}
```

Figura 25: Código CSS para navegación

Con este código CSS conseguiríamos reordenar los elementos de la lista sin orden <ul> horizontalmente, dando el aspecto de menú rápido.

El resultado final sería el mostrado en la figura 26:

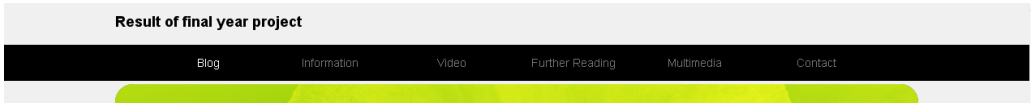

Figura 26: Resultado final de título y navegación

Introducción

Esta simple sección sólo mostrará una imagen de bienvenida a la página con el título del proyecto "HTML5 support for accesible Media Player", junto con un texto introductorio a la página.

El código HTML correspondiente a esta sección es el mostrado en la figura 27:

```
<section id="intro">
            <header>
                  <h2>HTML 5 support for accessible Media
Player</h2>
            </header>
            <p>Multimedia content covers the Web, and we
should provide access to all people. For this reason, it is
very important to take into account accessibility
requirements in the player to avoid barriers and to ensure
access to this multimedia content as well as their resources.
One of the most frequent barriers is the technological
obstacle: the necessity for the user to install the required
plug-ins in to order to access video. The new standard HTML5
provides a solution to this problem. However, it does not
fully support accessibility requirements of W3C standards,
including WCAG and interaction requirement of UAAG.</p>
</section>
```

Figura 27: Código HTML para la introducción

Cómo se aprecia en el código anterior la imagen no está en HTML, sino que por comodidad se decidió referenciarla a través de CSS para poder añadir atributos y características propias diseñadas para la página.

En la figura 28 se observa la manera de introducir una imagen en una sección de forma simple, y la de otorgar atributos y características propias a través de CSS.

```
#intro {
  margin-top: 66px;
  padding: 44px;
  background: #467612 url("intro_flower.png") repeat-x;
}
```

Figura 28: Código CSS para la introducción

El resultado final sería el mostrado en la figura 29:

Figura 29: Resultado final de la introducción

Contenido principal

Éste es el contenido principal de la página web, diseñada para el proyecto.

Esta sección se podrá dividir en tres grupos:

- Contenido multimedia e información principal

    - Información de la página web directamente relacionada con el propósito del proyecto.
    - Video en HTML5 con subtitulado y audiodescripción.
- Comentarios
- Formulario

Contenido multimedia e información principal

En esta sección se introducirá al usuario al propósito de la página web, junto con una presentación llamativa acompañada de una imagen decorativa incluyendo la apariencia de un blog, con una cabecera.

En el texto incluido en esta sección, se hará una introducción a la página web con las normas y estándares sobre los que se apoya, para informar al usuario de cuáles se han estudiado y aplicado, y más concretamente las pautas de accesibilidad al contenido en la web (WCAG) y las pautas para la accesibilidad para agentes de usuario (UAAG), los cuales condicionan cómo se muestra el contenido multimedia en la web.

El código HTML de esta sección no es complicado y la estructura es sencilla, otorgándole forma de blog, con un título, fecha de creación, y el texto introductorio acompañado de una imagen para darle la apariencia de un post de blog. En la figura 30 se observa cómo es esta sección en código HTML.

```
<section id="principalblog">
      <article class="blogPost">
            <header>
                  <h3>Multimedia content must be accessible
for people with disabilities according to standards like the
Web Content Accessibility Guidelines (WCAG) and User Agent
Accessibility Guidelines (UAAG) of WAI</h3>
                  <p>Posted on <time datetime="2011-03-
10T23:31:45+01:00"> June 1st 2011</time> by <a
href="#">Alberto Sánchez-Heredero Pérez</a> - <a
href="#comments">2 comentarios</a></p>
            </header>
            <div id="texto">
                  <p>HTML5 offers access in the embedded
media player to be a huge step forward. The new standard
introduces the latest commands (such as &lt;video&gt; and
&lt;audio&gt;), which can create and label controls letting
keyboard shortcuts to access them, and screen readers to tell
the user which controls are available. Some HTML5 elements
provide support for some UAAG 2.0 guidelines.
                  ……p>
                  <p>The current version of HTML5 provides
some playback controls for the video: play/pause, full screen
toggle, volume and audio element controller toolbar. These
controls are very basic, not allowing the user to have full
control of the video. Following WCAG 2.0 and UAAG 2.0
guidelines, these native controls of HTML 5 are not enough.
It must have the following controls: end (stop), caption
on/off, search captions for text strings and select caption
language if closed captioning is available, audio description
on/off, rewind/forward seconds, volume up/ down, screen
reader full access and keyboard full access of controls.</p>
                  <img
            longdesc="http://www.w3.org/TR/html5/"src="CSS/im
            agenes/HTML5_Logo.png" id="flower"alt="Flower" />
                  <p>In order to build the controls as
complement HTML5 support for interaction requirements
included in UAAG 2.0, the accessible player have been
developed with combination of HTML5 and JavaScript technology
solution.</p>
            </div>
```

Figura 30: Código HTML para la información principal

Esta sección no tiene estilos relevantes en CSS, ya que es simplemente texto con cabeceras, pero para darle un formato diferenciado a esta última, se le cambiará ligeramente el estilo, como el tamaño de letra, tal y como se muestra en la figura 31.

```
h2 {
font-size: 28px;
line-height: 25px;
padding: 22px 0;
}
```

Figura 31: Código CSS para cabeceras h2

Contenido multimedia con vídeo en HTML5 con subtitulado y audiodescripción

Esta es la sección más importante de la página y del proyecto, y por lo tanto la más compleja en cuanto a código se refiere. En ella estarán contenidos todos los elementos indispensables que permiten la visualización del video, con sus controles incluidos, como los subtítulos y el audio para la audiodescripción.

Para reproducir vídeo en HTML5 usaremos la nueva etiqueta <video> que ha dado tanto que hablar y se comenta todo lo que hay que saber sobre esta etiqueta en el apartado 2.6.1.

Para asegurarnos que se puede reproducir en todos los navegadores que soporten esta etiqueta, será necesario el mismo vídeo codificado en distintos códecs para su correcta visualización. A continuación, dentro de la etiqueta estarán los atributos, tanto de HTML5 como de JavaScript, que necesitará la página web para poder responder a las funcionalidades que se han implementado, y que la mayoría responden a código JavaScript:

- Añadiendo el atributo nativo en HTML5 "preload", cargaremos el vídeo automáticamente cuando la página web se carge en el navegador. Así el usuario no tendrá que esperar a que se cargue el vídeo cuando le dé al "play".
- También se añadirá la función nativa de HTML5 "controls", para cargar los controles nativos de la etiqueta <video> en cuyos navegadores no esté activado JavaScript, y cuyos controles desactivaremos posteriormente a través de JavaScript para que den paso a los controles creados para este proyecto.
- La primera funcionalidad que está pensada para el video, es la opción de que el usuario pueda presionar el botón izquierdo del ratón en el vídeo y éste responda reproduciendo y pausando el contenido multimedia. Esto se consigue añadiendo simplemente como atributo *onclick="repro()"* a una función JavaScript, que se encargará de pausar o reproducir el vídeo dependiendo del estado en el que se encuentre.
- Pasando el puntero del ratón por encima del video, se mostrarán los controles que harán posible la correcta reproducción del mismo, añadiendo como atributo *onMouseOver="mostrarControles()"* a una función en JavaScript. Estos controles servirán para controlar el video, junto con la opción de seleccionar subtítulos y audiodescripción, que se explicarán más adelante.
- Por el contrario, si sacamos el ratón del video, los controles desaparecerán, dejando libre el vídeo para poder visualizarlo en su totalidad. Para que esto sea posible se ha añadido *onMouseOut="ocultarControles()"* como atributo a una función JavaScript.
- Para introducir la funcionalidad de controlar el vídeo por teclado, basta con añadir el atributo *onKeyDown="controlTeclado(event)"* en <video> para una vez que el usuario seleccione el vídeo para reproducir, cualquier tecla que presione que esté relacionada con los controles tendrá un efecto en el vídeo o en sus funciones, ya sea pausándolo, parándolo o añadiendo subtítulos y audiodescripción entre otros, y que más adelante se explicará con más detalle.
- Por último le añadiremos la anchura del vídeo con "width" y le otorgaremos un identificado con "id" para poder manipularlo posteriormente en JavaScript.

En la figura 32 se observa el código HTML5 para la etiqueta <video> anteriormente explicada:

```
<video width="660" id="player" preload controls
timeupdate="timeupdate()" onKeyDown="controlTeclado(event)"
onclick="repro()" onMouseOver="mostrarControles()"
onMouseOut="ocultarControles()" >
      <source src="21.mp4" type="video/mp4" />
      <source src="21.theora.ogv" type="video/ogg" />
      <source src="21.webm" type="video/webm">
</video>
```

Figura 32: Código HTML5 de <video>

Un código CSS para la etiqueta <video> no es necesario, ya que no se le aplica ningún estilo determinante para su estructura ni visualización. Los estilos que tiene son heredados de la etiqueta <body>, que solo aportan la colocación del vídeo en la página como contenido principal. El resultado final de la etiqueta <video> es el mostrado en la figura 33:

Figura 33: Resultado del vídeo en HTML5

Seguidamente se procederá a la explicación de las funciones JavaScript que hacen posible la reproducción del video, exclusivamente en la etiqueta <video>:

- Timeupdate(): esta función se explica más adelante en la sección “Introducción de subtítulos mediante JavaScript en HTML5”. Es la encargada de mostrar en pantalla el texto de los subtítulos en función del tiempo actual del video.
- onKeyDown="controlTeclado(event)": esta función es la encargada de detectar el momento en que el usuario pulsa una tecla que esté configurada como tecla rápida, y realiza las acciones sobre el vídeo que estén programadas, como se muestra en la siguiente figura 34.

```
function controlTeclado(evento){
      var control2=document.getElementById("volumeicon");
      mostrarControles();
      if (evento.keyCode==80){
                  repro();
      }
      if (evento.keyCode==69){
                  pararVideo();
      }
      if (evento.keyCode==82){
                  retroceder();
      }
      if (evento.keyCode==70){
                  avanzar();
      }
      if (evento.keyCode==85){
                  subirVolumen();
      }
      if (evento.keyCode==86){
            quitarAudioVideo();
      }
      if (evento.keyCode==68){
                  bajarVolumen();
      }
      if (evento.keyCode==73){
                  mostrarAyuda();
      }
      if (evento.keyCode==65){
                  quitarAD();
      }
      if (evento.keyCode==67){
                  ocultarSub();
      }
      if (evento.keyCode==77){
                  muteOrUnmute();
      }
      if (evento.keyCode==90){
                  cambiarAEsp();
      }

      if (evento.keyCode==88){
                  cambiarAEng();
      }
}
```

Figura 34: Código JavaScript de control por teclado

Cada tecla tiene un código numérico que puede ser distinguido por JavaScript, por tanto, se ha usado este funcionalidad para detectar la tecla que pulsa el usuario y aplicar una determinada acción de control sobre el video. Así, se puede controlar el vídeo a través del teclado sin tener que usar el ratón.

Estas funciones se explicarán más adelante, ya que son funciones directamente relacionadas con los controles creados para el vídeo y aquí sólo se realizan llamadas para reutilizar dichas funciones.

- Onclick=”repro()”: esta función sirve para reproducir o pausar el video, pulsando el botón izquierdo sobre el mismo. Dependiendo del estado en el que se encuentre el video, realizará una acción u otra, ampliando su control a pausar o reproducir también la audiodescripción, ya que depende directamente del minuto y duración en que se encuentre el video.

Así, si el vídeo se pausa la audiodescripción también, y si se reproduce después de pausado los dos reinician desde el último segundo que se estaba reproduciendo. Lo que nunca va a ocurrir, es que el vídeo esté pausado y la audiodescripción, ya que el código está preparado para que esto no ocurra, tal y como se muestra en la figura 35.

```
function repro(){
	if (video2.paused==true){
		video2.play();
		iniciar_proceso();
		document.getElementById("mu").play();
	}else{
		video2.pause();
		document.getElementById("mu").pause();
	}
	if(video2.ended==true){
		video2.play();
		iniciar_proceso();
		audio2.currentTime=0;
		document.getElementById("mu").play();
	}
}
```

Figura 35: Código JavaScript de reproducción del video

- onMouseOver="mostrarControles()": esta sencilla función se encarga de mostrar los controles creados para este proyecto, en el momento en el que el puntero del ratón se encuentre sobre el video. Es una forma de mostrar los controles, pensando en la idea de que cuando el usuario pase el puntero del ratón sobre el video, es lógico pensar que quiere usar los controles.

Por lo tanto, esta función se activará siempre y cuando el puntero del ratón se encuentre sobre el video. Para llevarlo a cabo, simplemente se modificará su estilo CSS de opacity con el valor “1”, haciéndolo visible. Se puede observar en la figura 36.

```
function mostrarControles(){
	var con=document.getElementById("controls");
	con.style.opacity=1;
}
```

Figura 36: Código JavaScript para mostrar los controles del video

- onMouseOut="ocultarControles()": esta función es similar a la anterior, pero realiza la acción contraria. Cuando el usuario mueva el puntero del ratón fuera de la zona del video, los controles se ocultarán.

Está pensado en que el usuario, cuando haya realizado cualquier acción sobre los controles o directamente sobre el video, como se ha explicado anteriomente, éstos controles se oculten, dejando el vídeo completamente sin ningún elemento que se interponga en su visionado, como se observa en la figura 37.

```
function ocultarControles(){
      var con=document.getElementById("controls");
      con.style.opacity=0;
}
```

Figura 37: Código JavaScript para ocultar los controles del video

Como diversos reproductores en la web, también se ha añadido un botón en el centro del vídeo con la finalidad en que la acción de reproducirlo sea lo más intuitivo posible.

Para ello se ha introducido un botón que ocupe la totalidad del video, con un símbolo reconocible que da a entender que para reproducir el video, basta con seleccionar con el botón izquierdo dicho símbolo. El sencillo código HTML es el mostrado en la figura 38.

```
<button id="playGrande" class="botonNO"
```

Figura 38: Código HTML5 del botón de reproducción inicial

Con el código CSS de la figura 39 colocaríamos el botón sobre el video, para que una vez se cargue la página, sólo se vea el botón.

```
#playGrande{
      position:absolute;
      background:url("playg.png") center no-repeat;
      text-align:center;
      height:430px;
      width:660px;
      border:none;
      top:1190px;
      z-index:1;
      display:block;
}
```

Figura 39: Código CSS del botón de reproducción inicial

El resultado del botón de reproducción inicial sería el mostrado en la figura 40:

Figura 40: Resultado final del botón de reproducción inicial

Y con el código JavaScript de la figura 41 comenzaría la reproducción del video, reutilizando la función repro() y mostrando los controles que a continuación se explicarán.

```
function empezarVideo(){
      var play=document.getElementById("playGrande");
      var controles=document.getElementById("controls");

      controles.style.display="block";
      play.style.display="none";
      repro();
}
```

Figura 41: Código JavaScript para la reproducción del video inicial

Con estos controles se cumpliría el requisito 1, ya que se aporta tanto acceso por teclado como por ratón.

Controles de <video> en HTML5

A continuación se procederá a explicar el código JavaScript que permite que los atributos introducidos en <video> funcionen correctamente.

Todos están incluidos en una etiqueta para separarlo del resto del código y así tratarlos como un conjunto, como se muestra en la figura 42.

```
<div id="controls"
onMouseOut="ocultarControles()"onMouseOver="mostrarControles(
)" onKeyDown="controlTeclado(event)">
```

Figura 42: Código HTML5 para etiqueta de controles

Desde este punto del código están los elementos que forman los controles creados para este proyecto. Se enumerarán uno a uno y explicando su función:

- Barra de progreso: este elemento sirve para mostrar el estado actual del video, mientras se está reproduciendo. Para que funcione se ha combinado CSS y JavaScript, con la estructura que se observa en la figura 43.

```
<div id="barra" onmouseover="tiempo(event)"title="00:00">
      <div id="vacio" onclick="buscar(event)">
            <div id="div_completado">
                  <span></span>
            </div>
      </div>
</div>
```

Figura 43: Código HTML5 para la barra de progreso

Está divido en una serie de sección para poder controlarlo independientemente unos de otros. En la etiqueta “vacío”, se irán rellenado con JavaScript y CSS con un color de izquierda a derecha, dando la impresión de que avanza en función de la duración y el tiempo actual que se está reproduciendo el video.

El código CSS que se usa en la barra es el mostrado en la figura 44.

```
div#vacio
{
	background-color: #e4e4e4;
	border: 1px solid black;
	width: 650px;
	padding: 0px;
	padding-top: 0px;
	padding-left: 0px;
	padding-right: 0px;
	padding-bottom: 0px;
	height:12px;
	text-align:left;
	-moz-border-radius:5px;
	-webkit-border-radius:5px;
	-moz-border-radius:5px;
	border-radius:5px;
}

div#div_completado
{
	position: relative;
	top: 0px;
	left: 0px;
	background-color: #9af;
	width: 0px;
	padding-top: 5px;
	padding: 0px;
	z-index:1;
	color:#9af;
	height:12px;
	-moz-border-radius:5px;
	-webkit-border-radius:5px;
	border-radius:5px;
}
```

Figura 44: Código CSS de la barra de progreso

El código JavaScript que hace posible el efecto que la barra se mueva según avanza el vídeo, se puede observar en la figura 45.

```
function aumenta_barra(){
	porcentaje = Math.floor(((video2.currentTime * 1000) /
	milisec_barra) * 100);
	document.getElementById("div_completado").style.width =
	(porcentaje / 100) * tam_barra + "px";
	setTimeout("aumenta_barra();", 100);
}
```

Figura 45: Código JavaScript para animación de la barra de progreso

Con este código lo que se hace simplemente es aumentar el ancho de la división “div_completado”, según avanza la reproducción del video, siendo “milisec_barra” la duración del vídeo en milisegundos y “tam_barra” el tamaño que ocupa la barra en el navegador en

píxeles. Así, con setTimeout, hacemos una llamada sobre sí misma en el que cada llamada se actualiza el tiempo de reproducción del video, y por tanto la barra de progreso.

También se ha implementado la capacidad de que el usuario seleccione el tiempo actual de reproducción, pulsando el botón izquierdo del ratón sobre dicha barra y en el que una función JavaScript se encarga de reconducir el vídeo a ese tiempo, añadiendo en la etiqueta "vacio" el atributo *onclik="buscar(event)"*.

Esta función detecta las coordenadas donde se ha pulsado el botón izquierdo del ratón, y calcula el minuto y segundo exacto donde correspondería esa coordenada con la duración del video. Si se modifica el estado del video, también se tiene que modificar el de la audiodescripción, ya que tienen que estar sincronizados.

El código que lleva a cabo esta acción se encuentra en la figura 46:

```
function buscar(evento){
    var coord;
    var cal;
    var barra=document.getElementById("barra");
    var valor= obtenerPosicionAbsoluta(barra);
    coord=evento.clientX-valor.left;
    if(video2.currentTime==0){
      document.getElementById("div_completado").style.width=c
oord+"px";
    }
    cal=Math.round((coord/tam_barra)*100);
    video2.currentTime=Math.round((cal*video2.duration)/100);
      audio2.currentTime=(cal*audio2.duration)/100;
}
```

Figura 46: Código JavaScript para la barra de progreso

Esta función se apoya en otra llamada "obtenerPosicionAbsoluta" que es la encargada de obtener el espacio en pixeles, que hay desde el margen izquierdo del navegador a la barra de progreso. Con este valor, podremos calcular la posición relativa del puntero en la barra, para posteriormente calcular el tiempo del video. La figura 47 muestra el código de esta función:

```
function obtenerPosicionAbsoluta(element) {
      if (typeof element == "string")
        element = document.getElementById(element)

      if (!element) return { top:0,left:0 };

      var y = 0;
      var x = 0;
      while (element.offsetParent) {
        x += element.offsetLeft;
        y += element.offsetTop;
        element = element.offsetParent;
      }
      return {top:y,left:x};
}
```

Figura 47: Código JavaScript de obtener posición absoluta

El resultado final para la barra de progreso sería el mostrado en la figura 48:

Figura 48: Resultado final para la barra de progreso

- Botón de reproducir/pausar: este botón simplemente se encarga de reproducir o pausar el vídeo según el estado en el que éste se encuentre, y cambiando la imagen de fondo mediante CSS. El código HTML5 de este botón se puede observar en la figura 49 y será muy similar a todos los botones de estos controles.

```
<div id="botonPlay" >
      <button title="Reproducir/Pausar
      video"onfocus="mostrarControles()" id="playpause"
      onclick="iniciar_proceso()">
            <div id="playpauseIcon"></div>
      </button>
```

Figura 49: Código HTML5 para el botón de reproducir/pausar

Para colocar este botón en los controles se necesitarán unos estilos en CSS determinados, no muy complicados pero sí los necesarios como para colocarlos exactamente donde queremos. Para ello necesitamos que dé la mayor apariencia posible de que todo es un conjunto, por lo que eliminaremos los bordes y haremos el fondo transparente, tal y como se ve en la figura 50.

```
#playpause {
      background:transparent;
      border-left:none;
      border-top:none;
      border-bottom:none;
      cursor:pointer;
      height:40px;
      width:45px;
}
```

Figura 50: Código CSS para el botón de reproducir/pausar

Las funciones JavaScript de este botón serán "mostrarControles()", explicada anteriormente y las necesarias para reproducir el video, como se observan en la figura 51:

```
video.addEventListener('play',function(e) {
document.getElementById('playpauseIcon').className="playpause
-pause";
}, true);

video.addEventListener('pause',function(e) {
      document.getElementById('playpauseIcon').className="pla
ypause-play";
}, true);

video.addEventListener('ended',function(e) {
      document.getElementById('playpauseIcon').className="pla
ypause-play";
}, true);

document.getElementById('playpause').addEventListener('click'
,function() {
if (video.paused) {
            video.play();
            document.getElementById("mu").play();
      } else {
            if (video.ended) {

      document.getElementById('player').currentTime=0;

      document.getElementById('playpauseIcon').className="pla
ypause-play";
                  video.play();
                  document.getElementById("mu").play();
            } else {
                  video.pause();
                  document.getElementById("mu").pause();
            }
      }
}, true);
```

Figura 51: Código JavaScipt para el botón de reproducir/pausar

Las tres primeras funciones simplemente cambian el atributo "class" del botón, para modificar la imagen del botón de "reproducir" a "pausar", en el que aunque el control se centra en <video>, la acción se realiza sobre el botón.

En la siguiente función se modifica el estado del vídeo según el estado en que éste se encuentre al pulsar el botón izquierdo del ratón sobre dicho botón, a la vez que la audiodescripción ya que tienen que estar sincronizados. Como se modifica el estado del vídeo de "play" a "pause" y viceversa, también se tiene que cambiar la imagen del botón con los símbolos de "play" y "pause" respectivamente.

El resultado final para el botón de reproducir/pausar es el mostrado en la figura 52:

Figura 52: Resultado final para el botón de reproducir/pausar

- Botón de parar video: este botón simplemente se encarga de parar el video, reiniciando el tiempo del vídeo y de la audiodescripción, usando una imagen estática ya que no se necesitará de ningún cambio cuando se realice esta acción. El código HTML5 es similar a los botones de los controles, como se puede observar en la figura 53:

```
<div id="botonStop">
      <button title="Parar
      video"onfocus="mostrarControles()"onclick="pararVideo()
      " id="stop">
            <div id="stopIcon"></div>
      </button>
```

Figura 53: Código HTML5 para el botón de parar video

El código CSS para estos botones como para el resto son prácticamente idénticos, con alguna diferencia en algunos detalles de bordes, como se aprecia en la figura 54.

```
#stop{
      background:transparent;
      cursor:pointer;
      height:40px;
      width:45px;
      border-top:none;
      border-bottom:none;
      border-left:none;
}
```

Figura 54: Código CSS para el botón de parar video

Las funciones JavaScript será parecidas a los anteriores incluyendo "mostrarControles()", pero con la diferencia en la acción particular que estos realicen, como en este caso "pararVideo()". Esta función simplemente para el vídeo y la audiodescripción reiniciando su tiempo actual a 0 y pausándolo para evitar que se reproduzcan automáticamente. La función se puede observar en la figura 55.

```
function pararVideo(){
      video2.pause();
      video2.currentTime=0;
      audio2.pause();
      audio2.currentTime=0;
}
```

Figura 55: Código JavaScript para parar el video

El resultado final para el botón de parar vídeo es el mostrado en la figura 56:

Figura 56: Resultado final para el botón de parar video

- Botón de retroceso del video: este botón permite retroceder el tiempo 5 segundos del vídeo a decisión del usuario, pulsando el botón izquierdo del ratón sobre dicho botón. El código HTML5 de dicho botón se observa en la figura 57.

```
<div id="botonRetroceso">
      <button title="Retroceder
      video"onfocus="mostrarControles()"onclick="retroceder()
      " id="retro">
      <div id="retroIcon"></div>
      </button>
</div>
```

Figura 57: Código HTML5 para el botón de retroceso del video

El código CSS para este botón es similar al anterior, como se aprecia en la figura 58:

```
#retro{
      background:transparent;
      cursor:pointer;
      height:40px;
      width:45px;
      border-top:none;
      border-bottom:none;
      border-left:none;
}
```

Figura 58: Código CSS para el botón de retroceso del video

Las funciones JavaScript son las mismas para los demás, con la excepción de la acción particular de dicho botón, en este caso la de retroceder el vídeo. Como al retroceder el tiempo del video, puede estar en cualquier minuto y segundo, hay que controlar las distintas posibilidades que pueden ocurrir y que no excedan los límites del video, siendo el minuto 0 el mínimo, y la duración del vídeo el máximo. Estas comprobaciones se pueden observar en el código JavaScript de la figura 59:

```
function retroceder(){
      if((video2.currentTime-5)<=0){
                  video2.currentTime=0;
      }else{
                  video2.currentTime=video2.currentTime-5;
      }
      if((audio2.currentTime-5)<=0){
                  audio2.currentTime=0;
      }else{
                  audio2.currentTime=audio2.currentTime-5;
      }
}
```

Figura 59: Código JavaScript para retroceder el video

El resultado final para el botón de retroceder vídeo es el mostrado en la figura 60:

Figura 60: Resultado final para el botón de retroceder video

- Botón de avance del video: este botón es similar al anterior pero realizando la acción contraria, permitiendo avanzar el tiempo del vídeo en 5 segundos a decisión del usuario, pulsando el botón izquierdo del ratón sobre dicho botón. El código HTML5 de dicho botón se observa en la figura 61.

```
<div id="botonAvance">
      <button title="Avanzar
      video"onfocus="mostrarControles()"onclick="avanzar()"
      id="avance"><div id="avanIcon"></div>
      </button>
</div>
```

Figura 61: Código HTML5 para avanzar el video

El código CSS es similar al anterior como se puede apreciar en la figura 62:

```
#avance{
      background:transparent;
      cursor:pointer;
      height:40px;
      width:45px;
      border-top:none;
      border-bottom:none;
      border-left:none;
}
```

Figura 62: Código CSS del botón de avance del video

Las funciones JavaScript son las que comparten los botones de los controles como "mostrarControles()" con la excepción de la función que realiza la acción particular asociado a este botón, en este caso avanzar el tiempo del video. Para realizar dicha acción se tienen que controlar que no excedan los límites del vídeo al igual que el anterior botón, comprobando que no baje del mínimo de tiempo que es 0 y del máximo que es la duración total del video. Esto se puede ver en la figura 63 y se activaría pulsándolo con el botón izquierdo del ratón en dicho botón de la barra de controles.

```
function avanzar(){
      if((video2.currentTime+5)>=video2.duration){
                  video2.currentTime=video2.duration;
      }else{
            video2.currentTime=video2.currentTime+5;
      }
      if((audio2.currentTime+5)<=0){
                  audio2.currentTime=audio2.duration;
      }else{
                  audio2.currentTime=audio2.currentTime+5;
      }
}
```

Figura 63: Código JavaScript para avanzar el video

El resultado final para el botón de avanzar vídeo es el mostrado en la figura 64:

Figura 64: Resultado final para el botón de avanzar video

- Tiempo de video: este elemento no son botones, sino texto en HTML que mediante javaScript se va moficando su valor según avanza el video. Tendrá dos tiempo, el tiempo actual de reproducción y el tiempo total del video, no pudiendo el primero ser mayor que el segundo. El código HTML5 de este elemento es el mostrado en la figura 65.

```
<div id="tiempo">
      <output id="display1">--:--</output>
      <a>/</a>
      <output id="display2">--:--</output>
</div>
```

Figura 65: Código HTML5 para el tiempo

El código CSS no tiene mayor dificultad que el darle estilos para colocarlos correctamente en el centro de la barra de controles, y diferenciar un tiempo de otro con un color diferente, como se aprecia en la figura 66.

```
#display1{
      color:white;
      padding-top:5px;
}

#tiempo{
padding-top:8px;
padding-left:15px;
padding-right:10px;
width:60px;
}
```

Figura 66: Código CSS para el tiempo

La función JavaScript de este elemento simplemente se encarga de calcular el tiempo actual y total del vídeo mediante métodos específicos de JavaScript, para después dividir dicho tiempo minutos y segundos, y actualizando dichos valores en su elemento correspondiente con innerHTML, como se observa en la figura 67.

```
video.addEventListener('timeupdate',function(e) {
      var s=e.target.currentTime;
      var h=Math.floor(s/3600);
      s=s%3600;
      var m=Math.floor(s/60);
      s=Math.floor(s%60);
      if (s.toString().length < 2) s="0"+s;
      if (m.toString().length < 2) m="0"+m;

      var duracion=e.target.duration;
      var min=Math.floor(duracion/60);
      var seg=duracion%3600;
      seg=Math.floor(seg%60);

  document.getElementById('display2').innerHTML= min+":"+seg;
  document.getElementById('display1').innerHTML = m+":"+s;

}, true);
```

Figura 67: Código JavaScript para el tiempo

El resultado final para el tiempo de vídeo es el mostrado en la figura 68:

Figura 68: Resultado final para el tiempo de video

- Control de volumen: está compuesto por varios elementos.
  - Botón de silenciar el volumen del video: este botón se encarga de silenciar completamente el vídeo junto con la audiodescripción y el audio del propio video, como se observa en la figura 69.

```
<div id="vol">
<button title="Silenciar
video"onfocus="mostrarControles()"id="audio" value="muted"
onClick="muteOrUnmute()">
<div id="volumeicon"></div>
</button>
</div>
```

Figura 69: Código HTML5 para silenciar el video

El código CSS para este botón es similar a los anteriores, con la diferencia de algunos detalles de bordes, como se aprecia en la figura 70.

```
#audio{
      background:transparent;
      cursor:pointer;
      height:40px;
      width:45px;
      border-top:none;
      border-bottom:none;
}
```

Figura 70: Código CSS del botón para silenciar el video

Comparte la función JavaScript de "mostrarControles()" como las anteriores, pero usa otra función que realiza la acción concreta de este botón que es "onclick=muteOrUnmute()", la cual silencia el audio del vídeo dependiendo del estado en que se encuentre, y modificando a su vez los elementos que están relacionados con el audio del vídeo para que el usuario observe que los audios no están disponibles. Esto se puede obervar en la figura 71, en cuyo código se aprecia que cambian los atributos "class" para modificar la imagen de fondo, y a su vez silencia la audiodescripción ya que tiene que estar sincronizados con el audio del video, y se entiende que si el usuario pulsa este botón con el botón izquierdo del ratón, quiere silenciar todos los audios del video, incluyendo dicha audiodescripción.

```
function muteOrUnmute() {
      video2.muted = !video2.muted;
      if (video2.muted==true){
            audio2.muted =true;
      }else{
            audio2.muted=false;
      }
      volmenActual=video2.volume;
      if(video2.muted==true){
    document.getElementById('volumeicon').className="volumen-
bajo";
    document.getElementById('ADIcono').className="ADIconoNO";
      document.getElementById('audioVideoIcon').className="AV
IconNO";
     audio2.muted;
      volumenWidth=document.getElementById("volumeLleno").sty
le.width;
   document.getElementById("volumeLleno").style.width=0+"px";
      }else{
    document.getElementById('volumeicon').className="volumen-
normal";
    document.getElementById('ADIcono').className="ADIconoSI";
      document.getElementById('audioVideoIcon').className="AV
IconSI";
      audio2.volume=0.7;
      video2.volume=volmenActual;
      document.getElementById("volumeLleno").style.width=(par
seFloat(volumenWidth))+"px";  }}
```

Figura 71: Código JavaScript para silenciar video

El resultado final para el botón de silenciar vídeo es el mostrado en la figura 72:

Figura 72: Resultado final para el botón de silenciar video

- Botón para bajar el volumen un 10%: este botón permite al usuario bajar el volumen del vídeo en un 10%. Es la alternativa a controlar el volumen desde un botón, a controlarlo con la barra de volumen que se explicará en el siguiente elemento. El código HTML5 de ese botón se puede observar en la figura 73:

```
<div id="icono">
<button title="Bajar volumen" id="icono1"
onclick="bajarVolumen()"></button>
</div>
```

Figura 73: Código HTML5 del botón para bajar el volumen 10%

El código CSS es muy parecido a los anteriores pero más simple, como se observa en la figura 74.

```
#icono1{
      background:url("icono_menos.png") no-repeat center
transparent;
      border:none;
      height:14px;
      width:14px;
}
```

Figura 74: Código CSS del botón para bajar volumen 10%

Este botón sólo tiene una función JavaScript, y es la que se encarga de bajar el volumen del video, controlando que no se sobrepasen los límites del volumen que es 0 para el mínimo, y 1 para el máximo. Esto se ve más fácilmente en la figura 75, y por lo que se aprecia, si se baja el volumen del vídeo también se baja el de la audiodescripción ya que controla el volumen global del video.

```
function bajarVolumen(){
      var control2=document.getElementById("volumeicon");
      var AD=document.getElementById("ADIcono");
      if((video2.volume-0.1)<=0){
      video2.volume=0;
      audio2.volume=0;
      document.getElementById("volumeLleno").style.width=0+"p
x";
      control2.className="volumen-bajo";
      AD.className="ADIconoNO";
      document.getElementById('audioVideoIcon').className="AV
IconNO";
      }else{
      video2.volume=video2.volume-0.1;
      audio2.volume=audio2.volume-0.1;
      v=parseInt(document.getElementById("volumeLleno").style
.width)-10;
      if(v<0){
      document.getElementById("volumeLleno").style.width=0+"p
x";
      }else{
      document.getElementById("volumeLleno").style.width=v+"p
x";
            }
      }
}
```

Figura 75: Código JavaScript para bajar el volumen del video un 10%

El resultado final para el botón de bajar el volumen del vídeo es el mostrado en la figura 76:

Figura 76: Resultado final para el botón para bajar el volumen del vídeo un 10%

- Botón para subir el volumen un 10%: este botón es igual al anterior, pero con la diferencia que realiza la acción contraria, sube el volumen global del vídeo un 10%. El código HTML5 de este botón es prácticamente idéntico al anterior, como se aprecia en la figura 77.

```
<div id="icono">
<button title="Subir volumen" id="icono2"
onclick="subirVolumen()"></button>
```

Figura 77: Código HTML5 del botón para subir el volumen un 10%

El código CSS para dar estilos a este botón es igual al anterior, solo cambia la imagen de fondo, tal y como se aprecia en la figura 78.

```
#icono2{
	background:url("icono_mas.png") no-repeat center
transparent;
	border:none;
	height:14px;
	width:14px;
}
```

Figura 78: Código CSS del botón para subir el volumen 10%

La función JavaScript de este botón es similar a la anterior, con la diferencia en que en lugar de bajar el volumen se sube, manteniendo el control de no salirse de los límites mínimos y máximos. Se puede observar en la figura 79.

```
function subirVolumen(){
	var control2=document.getElementById("volumeicon");
	var AD=document.getElementById("ADIcono");
	if((video2.volume+0.1)>1){
		video2.volume=1;
		audio2.volume=1;
	document.getElementById("volumeLleno").style.width=100+
"px";
	AD.className="ADIconoSI";
	document.getElementById('audioVideoIcon').className="AV
IconSI";
	}else{
		control2.className="volumen-normal";
		video2.volume=video2.volume+0.1;
		audio2.volume=audio2.volume+0.1;
		AD.className="ADIconoSI";
	document.getElementById('audioVideoIcon').className="AV
IconSI";
	v=parseInt(document.getElementById("volumeLleno").style
.width)+10;
	if(v>100){
		document.getElementById("volumeLleno").style.widt
		h=100+"px";
	}else{
		document.getElementById("volumeLleno").style.widt
	h=v+"px";
	}
	}
}
```

Figura 79: Código JavaScript para subir el volumen del video un 10%

El resultado final para el botón de subir el volumen del vídeo es el mostrado en la figura 80:

Figura 80: Resultado final para el botón para subir el volumen del vídeo un 10%

- Barra de control del volumen: esta barra permite seleccionar el volumen del video, de forma similar a la barra de progreso del video, con la diferencia en que ésta no se mueve y permanece estática mientras el usuario no interactúe con ella o pulse el botón de silenciar el vídeo como se ha visto anteriormente. El código HTML5 de dicha barra se puede observar en la figura 81.

```
<div id="volumeControl" title="Volumen">
<div id="volumeVacio" onclick="buscarVolumen(event)">
            <div id="volumeLleno">
                  <span></span>
            </div>
      </div>
</div>
```

Figura 81: Código HTML5 de la barra de control del volumen

El código CSS de esta barra es muy parecido al de la barra de progreso del vídeo (véase figura 44), ya que comparten muchos elementos y colores de fondo para la animación de ficha barra, como se aprecia en la figura 82.

```
div #volumeVacio
{
      background-color: #e4e4e4;
      border: 1px solid black;
      width: 100px;
      padding: 0px;
      padding-top: 0px;
      padding-left: 0px;
      padding-right: 0px;
      padding-bottom: 0px;
      height:12px;
      text-align:left;
      -moz-border-radius:5px;
      -webkit-border-radius:5px;
      border-radius:5px;
}
div #volumeLleno
{
      position: relative;
      top: 0px;
      left: 0px;
      background-color: #9af;
      width: 0px;
      padding-top: 5px;
      padding: 0px;
      z-index:1;
      color:#9af;
      height:12px;
      -moz-border-radius:5px;
      -webkit-border-radius:5px;
      border-radius:5px;
}
```

Figura 82: Código CSS para la barra de control del volumen

La función JavaScript también es similar a la de la barra de progreso del vídeo (véase figura 41). Se controla la coordenada en el eje de las X para posicionar el puntero en la barra, cada vez que el usuario pulse con el botón izquierdo del ratón sobre dicha barra. Así mediante un sencillo cálculo, se podrá calcular el volumen del vídeo con respecto al tamaño de la barra en píxeles en el navegador.

Esta función también controla los límites de la barra en píxeles. Si el usuario pulsa el botón izquierdo en una zona muy próxima al límite izquierdo, el volumen del vídeo será muy bajo, por lo que se ha optado por silenciar directamente tanto la audiodescripción como el audio del video, teniendo que cambiar las imágenes de dichos botones que se explicarán más adelante. Esto se puede observar en la figura 83.

```
function buscarVolumen(evento){
	var coord2;
	var control=document.getElementById("volumeLleno");
	var control2=document.getElementById("volumeicon");
	var valor = obtenerPosicionAbsoluta(control);

	coord2= evento.clientX-valor.left;
	if(coord2>100){
		coord2=100;
	}
	if(coord2<5){
		control2.className="volumen-bajo";
	document.getElementById('audioVideoIcon').className="AV
IconNO";
	document.getElementById('ADIcono').className="ADIconoNO
";
		video2.volume=0;
		audio2.volume=0;
	}else{
		control2.className="volumen-normal";
	document.getElementById('audioVideoIcon').className="AV
IconSI";
		if (audio2.muted==false){
	document.getElementById('ADIcono').className="ADIconoSI
";
		}
		video2.volume=0.5;
		audio2.volume=0.5;
	}
	if (video2.muted==true){
		video2.muted=!video2.muted;
	}
	control.style.width=coord2+"px";
	video2.volume=coord2/100;
	audio2.volume=coord2/100;
}
```

Figura 83: Código JavaScript para la barra de control de volumen

Para esta función también se usara la misma que se empleo para obtener la posición absoluta, la cual se puede ver en la figura 47.

El resultado final para la barra de control de volumen es el mostrado en la figura 84:

Figura 84: Resultado final para la barra de control de volumen

- Botón de habilitar/deshabilitar subtítulos: este botón permite al usuario habilitar o deshabilitar los subtítulos, sea cual sea su idioma. Esta función no supondría casi ninguna dificultad si solo tuviésemos un idioma de subtítulo, pero en este caso, como se aportan dos idiomas para el subtitulado, hay que controlar que cuando uno se muestre, el otro no y viceversa. El código HTML5 de este botón es el que se puede observar en la figura 85.

```
<div id="sub">
<button title="Habilitar/Deshabilitar
subtitulos"onfocus="mostrarControles()"id="botonsub"
value="ocultar" onclick="ocultarSub()">
<div id="subtitulo"></div>
</button>
</div>
```

Figura 85: Código HTML5 del botón para habilitar/deshabilitar subtitulos

El código CSS vuelve a ser muy similar al de anteriores botones con detalles de bordes, como se observa en la figura 86.

```
#botonsub {
      background:transparent;
      border-top:none;
      border-bottom:none;
      cursor:pointer;
      height:40px;
      width:45px;
}
```

Figura 86: Código CSS del botón para habilitar/deshabilitar subtítulos

La función JavaScript que permite habilitar y deshabilitar los subtítulos, como se ha explicado anteriormente, tiene que controlar que si el subtítulo de un idioma se ve, el otro no y viceversa, por lo que no es demasiado complicado pero se tiene que tener cuidado con no equivocarse a la hora de mostrarlos. Esto se arregla con una variable que indica si uno de ellos se muestra o no, cuando el momento de pulsar el botón izquierdo sobre dicho botón el que está ocultado se muestra y viceversa, y cambiando a su vez la imagen del botón para indicar si están habilitados o no los subtítulos con un simple vistazo. Para observar cómo funciona, se puede observar la figura 87.

```
function ocultarSub() {
      var sub1=document.getElementById("caption");
      var sub2=document.getElementById("caption2");
      var sub3=document.getElementById("caption3");
      var sub4=document.getElementById("caption4");
            if (ocultado==true){
            if(esp==true){
                  sub1.style.opacity=1;
                  sub2.style.opacity=1;
            }
            if(eng==true){
                  sub3.style.opacity=1;
                  sub4.style.opacity=1;
            }
      document.getElementById('subtitulo').className="ccSi";
            ocultado=false;
      }else{
            ocultado=true;
      document.getElementById('subtitulo').className="ccNo";
            if (esp==true){
                  sub1.style.opacity=0;
                  sub2.style.opacity=0;
            }
            if (eng==true){
                  sub3.style.opacity=0;
                  sub4.style.opacity=0;
            }
      }
}
```

Figura 87: Código JavaScript para habilitar/deshabilitar subtítulos

El resultado final para el botón de habilitar/deshabilitar subtítulos es el mostrado en la figura 88:

Figura 88: Resultado final para el botón de habilitar/deshabilitar subtítulos

- Botón para habilitar/deshabilitar audiodescripción: este botón es muy parecido al anterior, con la diferencia que en lugar de habilitar o deshabilitar los subtítulos, se haga lo mismo con la audiodescripción. El usuario puede realizar dicha acción con la audiodescripción opcionalmente, si quiere escuchar solo el video. El código HTML5 de este botón es el de la figura 89.

```
<div id="audioDes">
   <button title="Habilitar/Deshabilitar
Audiodescripción"onclick="quitarAD()"onfocus="mostrarControle
s()"id="botonAD">
      <div id="ADIcono"></div>
   </button>
</div>
```

Figura 89: Código HTML5 del botón para habilitar/deshabilitar audiodescripción

El código CSS de este botón es similar a la de anteriores botones, como se puede apreciar en la figura 90.

```
#botonAD{
      background:transparent;
      cursor:pointer;
      height:40px;
      width:45px;
      border-top:none;
      border-bottom:none;
      border-left:none;
}
```

Figura 90: Código CSS del botón para habilitar/deshabilitar audiodescripción

La función JavaScript que realizar la acción de este botón es la de silenciar la audiodescripción a petición del usuario pulsando dicho botón con el botón izquierdo del ratón. El código, como se observa en la figura 91, no es muy complicado ya que solo tiene que silenciar el archivo de audiodescripción y cambiar la imagen de fondo para indicar que está habilitado o deshabilitado con un solo vistazo.

```
function quitarAD(){
            audio2.muted = !audio2.muted;
            if (audio2.muted==false){
    document.getElementById('ADIcono').className='ADIconoSI';
            }else{
    document.getElementById('ADIcono').className='ADIconoNO';
            }
}
```

Figura 91: Código JavaScript del botón para habilitar/deshabilitar audiodescripción

El resultado final para el botón de habilitar/deshabilitar audiodescripción es el mostrado en la figura 92:

Figura 92: Resultado final para el botón de habilitar/deshabilitar audiodescripción

- Botón para habilitar/deshabilitar el audio del video: este botón realiza una acción parecida a la del anterior botón pero con la diferencia en que el elemento que modifica es el audio del video, dando la opción al usuario de poder escuchar solo la audiodescripción en lugar de dicho audio. El código HTML5 de este botón es el mostrado en la figura 93.

```
<div id="audioVideo">
     <button title="Habilitar/Deshabilitar audio de
video"onclick="quitarAudioVideo()"onfocus="mostrarControles()
"id="botonAudioVideo">
      <div id="audioVideoIcon"></div>
    </button>
</div>
```

Figura 93: Código HTML5 del botón para habilitar/deshabilitar audio del video

El código CSS de este botón es prácticamente idéntico al resto, como se puede apreciar en la figura 94.

```
#botonAudioVideo {
      background:transparent;
      border-top:none;
      border-bottom:none;
      border-left:none;
      cursor:pointer;
      height:40px;
      width:45px;
}
```

Figura 94: Código CSS del botón para habilitar/deshabilitar audio del video

La función JavaScript es muy parecía al anterior botón pero con la diferencia que el elemento que modifica es el vídeo directamente. Esto permite al usuario pulsando con el botón izquierdo del ratón sobre dicho botón, poder escuchar solo la audiodescripción, cambiando la imagen de fondo para poder diferenciar si el audio del vídeo está habilitado o no, tal y como se aprecia en la figura 95.

```
function quitarAudioVideo(){
      video2.muted = !video2.muted;
      volmenActual=video2.volume;
      if(video2.muted==true){
      document.getElementById('audioVideoIcon').className="AV
IconNO";
      volumenWidth=document.getElementById("volumeLleno").sty
le.width;
      }else{
      document.getElementById('audioVideoIcon').className="AV
IconSI";
            video2.volume=volmenActual;
      }}
```

Figura 95: Código JavaScript del botón para habilitar/deshabilitar audio del video

El resultado final para el botón de habilitar/deshabilitar audio del vídeo es el mostrado en la figura 96:

Figura 96: Resultado final para el botón para habilitar/deshabilitar audio del video

- Botón de ayuda para los controles por teclado: este botón permite al usuario mostrar una ayuda acerca de los controles por teclado, mostrado por un recuadro sobre el vídeo con información sobre las teclas que debe pulsar para realizar una determinada acción sobre el video. Estas acciones por teclado son las mismas que las que están en la barra de controles y que se ha explicado a lo largo de este apartado. El código HTML5 de este botón es el mostrado en la figura 97.

```
<div id="ayuda">
      <button title="Mostrar ayuda controles por
teclado"onclick="mostrarAyuda()"onfocus="mostrarControles()"i
d="botonAyuda">
<div id="ayudaIcon"></div>
</button>
</div>
```

Figura 97: Código HTML5 del botón de ayuda

El código CSS para este botón es similar a los anteriores, como se aprecia en la figura 98.

```
#botonAyuda {
      background:transparent;
      border-top:none;
      border-bottom:none;
      border-left:none;
      cursor:pointer;
      height:40px;
      width:45px;
}
```

Figura 98: Código CSS del botón de ayuda

La función JavaScript que hace posible la visualización de esta ayuda es muy simple. Basta con modificar el estilo del elemento de ayuda en el documento HTML para que sea visible cuando se pulse dicho botón con el botón izquierdo del ratón, y se oculte cuando se vuelva a pulsar, tal y como se muestra en la figura 99.

```
function mostrarAyuda(){
    if(ayuda==false){
    document.getElementById("infoAyuda").className='ayudaSI';
    ayuda=true;
    }else{
    document.getElementById("infoAyuda").className='ayudaNO';
    ayuda=false;
    }
}
```

Figura 99: Código JavaScript del botón de ayuda

El código HTML5 de ayuda que muestra esta función es el mostrado en la figura 100. Este elemento tiene como estilo de inicio que no se muestre cuando se cargue la página, estando oculto pero el código presente con un atributo “class” que lo oculta.

Así, si el navegador con el que entra el usuario a la página está desactivado, este elemento de ayuda no se muestra, evitando así que la página tenga un error bastante grave.

```
<div id="infoAyuda" class="ayudaNO">
<header>
            <h3>Controles de video por teclado</h3>
      </header>
      <ul>
            <li>Play/Pause: tecla p</li>
            <li>Parar: tecla e</li>
            <li>Retroceder 5 segundos: tecla r</li>
            <li>Avanzar 5 segundos: tecla f</li>
            <li>Subir volumen 10%: tecla u</li>
            <li>Bajar volumen 10%: tecla d</li>
            <li>Silencio audio de video: tecla v</li>
            <li>Silencio Audio descripción: tecla a</li>
      </ul>
      <ul>
            <li>Silencio video: tecla m</li>
            <li>Mostrar ayuda: tecla i</li>
            <li>Subtitulos Si/No: tecla c</li>
            <li>Subtitulos en español/spanish: tecla z</li>
            <li>Subtitulos en inglés/english: tecla x</li>
      </ul>
</div>
```

Figura 100: Código HTML5 del elemento de ayuda

El código CSS simplemente permite darle estilo para que su presentación al usuario y en el vídeo sea el correcto, como se observa en la figura 101.

```
#infoAyuda{
      position:absolute;
      top:1222px;
      opacity:0.7;
      width:660px;
}
.ayudaSI{
      display:block;
      background-color:#D8D8D8;
}
.ayudaNO{
      display:none;
}
```

Figura 101: Código CSS del elemento de ayuda

El resultado final para el botón de mostrar ayuda es el mostrado en la figura 102:

Figura 102: Resultado final para el botón de mostrar ayuda

- Botones de idioma de subtítulos: estos botones permite al usuario seleccionar el idioma de los subtítulos que se muestra en tiempo real sobre el video. El usuario podrá cambiar el idioma de los subtítulos en cualquier momento, con solo pulsar dichos botones con el botón izquierdo del ratón o, por el contrario, a través del teclado.

El código HTML5 de estos botones no es muy diferente del resto, como se aprecia en la figura 103.

```
<div id="idioma">
      <button title="Subtitulos en español"id="banderaEsp"
      onclick="cambiarAEsp()"></button>
</div>

<div id="idioma">
      <button title="Subtitulos en inglés"id="banderaEng"
      onclick="cambiarAEng()"></button>
</div>
```

Figura 103: Código HTML5 de los botones de idiomas de subtitulos

El código CSS de estos botones no tiene nada especial, salvo los estilos necesarios para colocarlos correctamente en la página y las imágenes de fondo para distinguir dichos botones, como se ve en la figura 104.

```
#idioma{
      padding-left:10px;
}

#banderaEsp {
      position:absolute;
      background:url("spain.png") no-repeat transparent;
      border:none;
      height:17px;
      width:25px;
      top:17px;
}

#banderaEng {
      position:absolute;
      background:url("uk.png") no-repeat center transparent;
      border:none;
      height:17px;
      width:25px;
      top:36px;
}
```

Figura 104: Código CSS de los botones de idiomas de subtítulos

La función JavaScript de la figura 105 de estos botones es diferente una de otra, ya que una vez se pulse uno de ellos, el idioma al que corresponda tiene que aparecer y el otro desaparecer, por lo que hay que llevar un cierto control mediante variables booleanas que nos indicarán el estado de los subtítulos.

En el momento en que se pulse cualquiera de estos botones de idioma, aparecerán automáticamente si existe el caso de que anteriormente se haya deshabilitado dichos subtítulos. Así, se entiende que si el usuario pulsa estos botones, es que quiere ver dichos subtítulos por lo que la opción de mantenerlo ocultos no sería lógica.

```
function cambiarAEsp(){
	var sub1=document.getElementById("caption");
	var sub2=document.getElementById("caption2");
	var sub3=document.getElementById("caption3");
	var sub4=document.getElementById("caption4");
		esp=true;
		eng=false;

		sub1.style.opacity=1;
		sub2.style.opacity=1;
		sub3.style.opacity=0;
		sub4.style.opacity=0;

	document.getElementById('subtitulo').className="ccSi";
	ocultado=false;
}

function cambiarAEng(){
	var sub1=document.getElementById("caption");
	var sub2=document.getElementById("caption2");
	var sub3=document.getElementById("caption3");
	var sub4=document.getElementById("caption4");
		eng=true;
		esp=false;

		sub1.style.opacity=0;
		sub2.style.opacity=0;
		sub3.style.opacity=1;
		sub4.style.opacity=1;
	document.getElementById('subtitulo').className="ccSi";
		ocultado=false;
}
```

Figura 105: Código JavaScrpipt de los botones de idiomas de subtítulos

El resultado final para los botones de selección de idioma para los subtítulos es el mostrado en la figura 106:

Figura 106: Resultado final para los botones de idioma de subtítulos

Hasta aquí llega la explicación de la barra de controles. El resultado final de dicha barra se pude observar en la figura 107.

Figura 107: Resultado final de la barra de controles

Una vez terminada la explicación de los controle del vídeo en HTML, es conveniente que se explique en funcionamiento de la audiodescripción ya que es un elemento fundamental del video.

El elemento de audiodescripción es simplemente una etiqueta <audio> introducida en la página web, enlazado con un archivo de audio en diferentes formatos para ser soportado por la mayoría de navegadores. El código HTML5 es el mostrado en la figura 108, y no es más que lo ya explicado en el apartado 2.6.2.

```
<audio id="mu">
      <source src="nadia.wav" type="audio/wav" />
      <source src="nadia.mp3" type="audio/mp3" />
      <source src="nadia.ogg" type="audio/ogg" />
</audio>
```

Figura 108: Código HTML5 de la audiodescripción

El código CSS de la figura 109 simplemente lo oculta en el navegador para conseguir que el archivo esté en el código pero no sea visible para el usuario ya que no tiene ninguna utilidad para él.

```
#mu{
      display:none;
}
```

Figura 109: Código CSS de la audiodescripción.

Este elemento no necesita de JavaScript, ya que todas las acciones que sea realizan en él ya se ha explicado anteriormente con los botones que está relacionados con la audiodescripción. Con todos estos controles, se cumplirían los requisitos 2 y 3, ya que se ha creado una interfaz accesible y el usuario puede elegir la forma en la que accede a la información y en la que se muestra el contenido, aportando subtítulos en varios idiomas y opciones de audiodescripción.

Menú lateral

Para este simple menú, se ha usado una de las etiqueta principales que se han creado especialmente para HTML5, <aside>. Ésta etiqueta que ya fue mencionada en el apartado 2.5.1, representa una sección de la página que abarca un documento tangencialmente relacionado con el contenido que lo rodea, por lo que se le puede considerar un contenido independiente.

En esta página, se usado esta etiqueta para añadir un menú lateral con acceso a otros contenido de la web, tales como páginas de interés relacionadas con el proceso de creación de la página en HTML5.

El código en HTML5 de esta sección se puede observar en la figura 110.

```
<aside id="menu">
      <section id="fr">
            <header >
                  <h2>Further reading</h2>
            </header>
            <ul>
            <li><a
href="http://dev.w3.org/html5/spec/Overview.html">W3C. HTML5.
A vocabulary and associated APIs for HTML and XHTML. Working
Draft 05 April 2011.</a></li>
            <li><a
href="http://www.w3.org/WAI/intro/wcag.php">W3C, WAI, Web
Content Accessibility Guidelines (WCAG), 2011.</a></li>
            <li><a href="http://www.w3.org/WAI/">W3C, Web
Accessibility Initiative (WAI), 2011.</a></li>
            <li><a
href="http://www.w3.org/WAI/intro/uaag.php">W3C, WAI, User
Agent Accessibility Guidelines (UAAG), 2011.</a></li>
            <li><a href="http://html5accessibility.com/">The
Paciello Group, 2011, New HTML5 feature accessibility support
in Windows Browsers.</a></li>
            <li><a
href="http://terrillthompson.blogspot.com/2010_08_01_archive.
html">Terrill Thompson, 2010, Keyboard and Screen-reader
Accessibility</a></li>
            <li><a href="http://diveintohtml5.org/table-of-
contents.html">Mark Pilgrim, Dive Into HTML5, What Works on
the Web</a></li>
            <li><a href="http://john.foliot.ca/accessibility-
and-html5-today/">John Foliot , 2011, &lt;video&gt;,
Accessibility and HTML5 Today”</a></li>
            <li><a
href="http://webaxe.blogspot.com/2011/03/accessible-youtube-
html5-video.html">Accessible YouTube & HTML5 Video.</a></li>
            <li><a href="http://webaim.org/blog/future-web-
accessibility-html5-video/">Future Web Accessibility: HTML5
&lt;video&gt;, 2010.</a></li>
            <li><a
href="http://doi.acm.org/10.1145/1535654.1535679">Silvia
Pfeiffer and Conrad Parker. 2009. Accessibility for the HTML5
\<video\> element. In Proceedings of the 2009 International
Cross-Disciplinary Conference on Web Accessibililty (W4A)
(W4A '09). ACM, New York, NY, USA, 98-100.
DOI=10.1145/1535654.1535679</a></li></ul>
      </section>
      <section id="mul">
            <header >
                  <h2>Multimedia</h2>
            </header>
            <ul><li><a
href="http://html5videoguide.net/presentations/WebVTT/#contro
```

Figura 110: Código HTML5 de la etiqueta <aside>

Los estilos dados a esta sección no son muy complicados, sino los suficientes para darle un estilo moderno y actual.

Para ello se ha optado por usar elementos de CSS3, que aunque no eran los objetivos de este proyecto, se ha visto la oportunidad de probar algunas nuevas características de la evolución de CSS2 y que como viene siendo habitual con los nuevos estándares, no todos los navegadores lo soportarán, como se verá en el apartado 5.4.

Estas características nuevas son muy pocas para evitar saturar a la página de elementos que no todos los navegadores soportarán, pero que se explicará claramente en qué navegadores se podrá usar y en cuáles no.

El código CSS de esta sección se puede observar en la figura 111.

```
aside {
    display: table-cell;
    width: 300px;
    background-image:-moz-linear-gradient(right, #f0f0f0
30%,#D8D8D8 150%);
}
aside section {
      margin: 22px 0 0 22px;
      padding: 11px 22px;
      background-image:-moz-linear-gradient(top,  #D8D8D8 8%,
#f0f0f0 90%);
      -moz-border-radius: 11px;
      -webkit-border-radius: 11px;
}
aside section ul {
            margin: 0px;
            padding-left:0px;
}
aside section ul li a {
      display: block;
      text-decoration: none;
      color: #000;
}
aside section ul li a:hover {
      text-decoration: underline;
}
```

Figura 111: Código CSS para la etiqueta <aside>

Las funciones JavaScript aquí son inexistentes, ya que esta sección sólo sirve para mostrar una sección que representa un menú lateral en la página.

El resultado final de esta sección se observa en la figura 112.

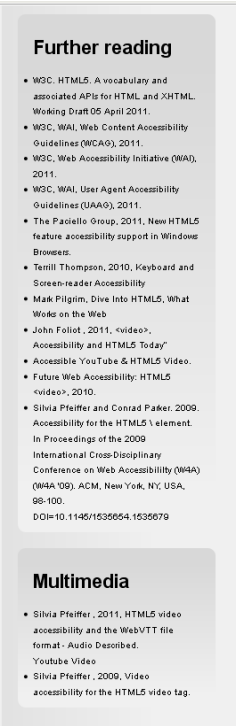

Figura 112: Resultado final de la etiqueta <aside>

Pie de página

Esta es la última sección de la página creada para este proyecto. En ella se ha creado un pie de página en HTML5, que se podía hacer perfectamente en la anterior versión 4.01 pero en esta ocasión con una nueva etiqueta creada para el nuevo estándar como es <footer>.

Esta etiqueta, como se vio en el apartado 2.5.1, representa el pie de una sección con información acerca de la página, como el autor y profesor asistente con un enlace a su página personal.

El código HTML5 de esta sección se puede observar en la figura 113.

```
<footer id="pie">
      <div>
            <section id="Blogroll">
            <ul><p>Player developed by Alberto Sanchez-
      Heredero, It is included in his Undergraduate Thesis
      Project, supervised by <a
      href="http://labda.inf.uc3m.es/doku.php?id=es:labda_per
      sonal:personal_lmoreno"class="azul">Lourdes Moreno</a>
      (Assistant Professor) of <a class="morado">Universidad
      Carlos III de Madrid.</a></p></ul>
            </section>
            <section id="popular">
            <ul><p>LaBDA, Advanced Databases Group Computer
science department Polytechnic School <a
class="morado">Universidad Carlos III de Madrid.</a></p></ul>
            </section>
      </div>
</footer>
```

Figura 113: Código HTML5 del pie de página

El código CSS de este pie de página es similar a la sección anterior, ya que se limita a darle estilos para darle una estructura legible y bien presentada, pero solo con CSS2 a diferencia de la sección <aside>. El código CSS del pie de página es el que observa en la figura 114.

```
footer {
	position: absolute;
	left: 0;
	width: 100%;
	background: #222;
}
footer div {
	display: table;
	margin: 0 auto;
	padding: 44px 0;
	width: 940px;
	color: #777;
}
footer div section {
	display: table-cell;
	width: 300px;
}
footer div #about, footer div #blogroll {
	padding-right: 20px;
}
footer h3 {
	color: #FFF;
}
footer a {
	color: #999;
}
footer a:hover {
	color: #FFF;
	text-decoration: none;
}
footer ul {
margin: 0 0 0 40px;
	list-style: square;
	color: #565656;
}
footer ul li a {
	display: block;
}
```

Figura 114: Código CSS del pie de página

Al igual que la sección <aside>, no se necesita de código JavaScript ya que sólo se limita a mostrar un pie de página con texto y estructurado con CSS.

El resultado final se puede observar en la figura 115.

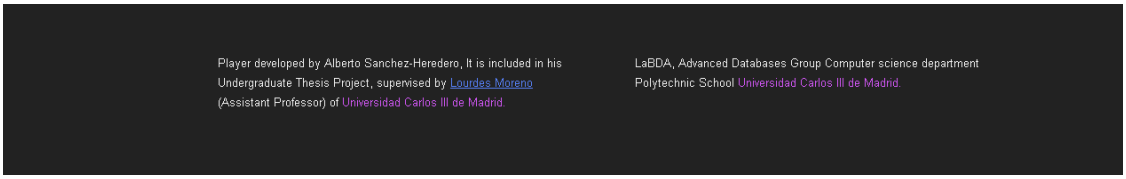

Figura 115: Resultado final del pie de página

Hasta ahora se ha explicado paso a paso y sección a sección, todos los elementos que forman la página en HTML5, y más en concreto los controles y los archivos multimedia que forma el objetivo de este proyecto. Pero todos estos elementos no tendrían sentido sin la característica fundamental mencionada en el título de este proyecto, que es el subtitulado.

A continuación se va a explicar la forma de la que se ha llevado a cabo esta funcionalidad, y la forma que se optado para que funciona correctamente y la de proporcionar estos subtítulos en varios idiomas.

Introducción de subtítulos mediante JavaScript en HTML5:

Para la introducción de subtítulos en un vídeo en HTML5, se va a proceder a la aplicación del lenguaje de JavaScript asociado a nuestra página.

Para ello, una opción sería incluir el texto en el propio documento HTML, pero el inconveniente más claro es el inmenso espacio que tengamos que ocupar para videos considerablemente largos y por tanto la cantidad de líneas de código en HTML que ocuparía esta información sin ninguna utilidad para el usuario, por lo que la opción más cómoda en cuánto a espacio y facilidad de manipulación, es la de crear un archivo XML y extraer la información de ese archivo para después usarlo más fácilmente en JavaScript y el DOM que éste incorpora.

La estructura del archivo XML será lo más sencilla posible, para evitar confusiones a la hora de una posible ampliación de los subtítulos, o más aún, de una posible reutilización del propio archivo para futuros videos. Esta estructura será la misma tanto para el archivo XML que contenga el texto en español como en inglés.

El ejemplo de la figura 116 muestra la estructura de un archivo que contiene subtítulos, siendo “data-begin” el segundo donde debería aparecer, “data-end” donde debería desaparecer, “class” el color del subtitulo, y “línea” la línea donde va a aparecer en el caso que haya dos frases a la vez, y que más tarde JavaScript se encargará de relacionarlo con el tiempo de reproducción del vídeo actual.

El nombre de las etiquetas es indiferente, ya que a la hora de obtener la información, le diremos a la función que usaremos en JavaScript, dónde está el texto que queremos que muestre.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<subtitulos>
<span data-begin="tiempo de inicio" data-end="tiempo de fin"
class=”color” línea=”línea 1 ó 2">*TEXTO*</span>
<span data-begin="tiempo de inicio" data-end="tiempo de fin"
class=”color” línea=”línea 1 ó 2">*TEXTO*</span>
<span data-begin="tiempo de inicio" data-end="tiempo de fin"
class=”color” línea=”línea 1 ó 2">*TEXTO*</span>
            ⋮
</subtitulos>
```

Figura 116: Fichero XML para introducir subtítulos

El siguiente paso sería relacionar este archivo XML a través de JavaScript, para lo que usaremos la llamada XMLHttpRequest y sus correspondientes métodos para poder acceder a la información y poder manipularla. Como se ha añadido la característica de varios idiomas a elegir en los subtítulos, se ha tenido que cargar dos XML de datos a la vez, el archivo XML de subtítulos en español y el del inglés.

El conjunto de métodos que permiten esto es el mostrado en la figura 117:

```
If(){
(para Firefox, Opera, Chrome, Safari, etc…)
var xml1=new XMLHttpRequest();
var xml2=new XMLHttpRequest();}else{
(para IE)
var xml1=new ActiveXObject("Microsoft.XMLHTTP");
      var xml2=new ActiveXObject("Microsoft.XMLHTTP");
}

xml1.open("GET","subnicoEsp.xml",false);
xml1.send();
xml2.open("GET","subnicoEng.xml",false);
xml2.send();
var xmlDoc1 = xml1.responseXML;
var xmlDoc2 = xml2.responseXML;
```

Figura 117: Código JavaScript para leer un archivo XML

Una vez hechas estas instrucciones, las últimas instrucciones guarda un objeto que podremos manipular para obtener la información que queremos, guardándolo en variables anteriormente declaradas en el archivo de JavaScript para separarlo en distintos campos dependiendo del tipo de información que queremos obtener, y de recorrerlos todos en un bucle for, como muestra la figura 118 y que se explica a continuación.

Cuando se identifica un elemento de subtitulo, antes de guardarlo se comprueba a qué línea está asignada con dos “if” anidados, y si es en la primera se añade en la cola de una variable array global “captions”, y si no, se añade a otra variable similar “captions2” para los subtítulos de la segunda línea, para después poder usarlo en la función siguiente que se encarga de mostrarlos en el vídeo. Una vez que está guardado en sus correspondiente array de texto, se añade el color correspondiente del subtitulo en un array llamado “colores” uno para cada array “captions”, el cual, cada vez que un subtítulo se muestre por pantalla, se añadirá como atributo del en el HTML, el color que está guardado en el array correspondiente, para que se encargue el fichero CSS de cambiarle de color.

Para asegurarse de que se recorren bien todos los arrays “captions” de subtítulos, se guarda en una simple variable numérica, su longitud para que no haya confusiones a la hora de mostrar el texto del subtítulo ni interfieran unos con otros.

```
var datos = xmlDoc1.getElementsByTagName("span");
var datos2 = xmlDoc2.getElementsByTagName("span");
var node = "", node2="";
var caption = "",caption2="";
var j=0, k=0;
for (var i = 0, node; node = datos[i]; i++) {
caption = {'start': parseFloat(node.getAttribute('data-
begin')),  'end': parseFloat(node.getAttribute('data-end')),
            'text': node.childNodes[0].nodeValue} ,
'li':node.getAttribute('linea')};
      if (caption.li==1){
            captions.push(caption);
            colores[k]=node.getAttribute('class');
            k++;
      }else{
            captions2.push(caption);
            colores2[j]=node.getAttribute('class');
            j++;
      }

}
for (var i = 0, node2; node2 = datos2[i]; i++) {
      caption2 = {'start':
parseFloat(node2.getAttribute('data-begin')), 'end':
parseFloat(node2.getAttribute('data-end')),
            'text': node2.childNodes[0].nodeValue,
'li':node2.getAttribute('linea')};
      if (caption2.li==1){
            captions3.push(caption2);
            k++;
      }else{
            captions4.push(caption2);
            j++;
      }
}
lineas = captions.length;
lineas2=captions2.length;
```

Figura 118: Código JavaScript para leer subtítulos de una archivo XML

Se vuelve a repetir la misma operación para guardar la información de los subtítulos en inglés, guardando en otra variable array global “captions3” para la primera línea, y “captions4” para la segunda línea de subtitulado. La diferencia con este mismo código es que no se guardan los colores de los subtítulos, ya que al ser la misma información pero con letra distintas, se reutiliza la información captada en la figura 117 por los arrays de colores, “colores” y “colores2”. Las líneas tampoco es necesario recontarlas, ya que sigue siendo el mismo número de frases de subtítulos y por tanto no es necesario contarlas.

Observando el código, se aprecia que los elementos de donde se extraen los datos en el “for” no es el mismo, en el primero se usa “datos” que hace referencia al primer objeto con el que se abre el archivo XML en español, y en el segundo se usa “datos” que hace referencia al segundo objeto con el que se hace referencia al archivo XML en inglés.

Para que no haya celdas de arrays vacías y que coincidan cada array caption con su correspondiente array de colores, se ha optado por crear dos contadores independientes para cada uno, llamándoles simplemente j y k, en el sólo avanzarán si su correspondiente array de subtítulos lo hace.

El siguiente paso mostrado en la figura 119, una vez guardada la información de los subtítulos en variables de JavaScript, es mostrarlo en el documento HTML. Para ello se encarga la siguiente función, que obtiene el tiempo exacto de reproducción del video, y lo compara con la información obtenida de los subtítulos, en el que supuestamente debería aparecer y desaparecer el subtítulo y lo muestra por pantalla con "innerHTML". Como contamos con dos arrays de subtítulos, necesitamos dos "for" que los recorran con outputs independiente y usando su array "colores" correspondiente, introduciéndolo en el HTML como un atributo "class" para poder identificarlo y cambiar el color posteriormente mediante CSS.

```
function timeupdate() {
var now = video.currentTime;
      var text = "", cap = "", cap2="", text2="", cap3="",
text3="", cap4="", text4="";
      for (var i = 0; i < lineas; i++) {
            cap = captions[i];
            cap3 = captions3[i];
            if (now >= cap.start && now <= cap.end) {
                  text = cap.text;
                  text3=cap3.text;
                  output.className=colores[i];
                  output3.className=colores[i];
                  break;
            }
      }
      for (var j = 0; j < lineas2; j++) {
            cap2 = captions2[j];
            cap4 = captions4[j];
            if (now >= cap2.start && now <= cap2.end) {
                  text2 = cap2.text;
                  text4= cap4.text;
                  output2.className=colores2[j];
                  output4.className=colores2[j];
                  break;
            }
      }
output.innerHTML = text;
output2.innerHTML= text2;
output3.innerHTML= text3;
output4.innerHTML= text4;
}
```

Figura 119: Código JavaScript para mostrar subtítulos en un video

Como ya hemos mencionado anteriormente, se hicieron dos secciones repetidas de código para separar la información del subtitulado en español y por otro lado en inglés. Por lo tanto, para mostrarlo en el documento HTML explicado en el párrafo anterior, es necesario que se repita de nuevo el mismo proceso.

El último paso sería posicionar el texto en el vídeo dependiendo de la anchura del mismo y su posición en la página web. Esto se consigue aplicando una serie de estilos mediante CSS como en la figura 120 y 121.

```
#caption {
 position: absolute;
 font-family: sans-serif;
 font-weight: bold;
 font-size:150%;
 text-shadow: black 1px 1px 3px;
 z-index:1;
 /*top:1690px;*/
 top:330px;
 width:660px;
 text-align:center;
 opacity:1;
}
 #caption2 {
 position: absolute;
 font-family: sans-serif;
 font-weight: bold;
 font-size:150%;
 text-shadow: black 1px 1px 3px;
 z-index:1;
 /*top:1715px;*/
 top:355px;
 width:660px;
 text-align:center;
 opacity:1;
}
```

Figura 120: Ejemplo de CSS para un subtitulo(1)

```
#caption3 {
 position: absolute;
 font-family: sans-serif;
 font-weight: bold;
 font-size:150%;
 text-shadow: black 1px 1px 3px;
 z-index:1;
 /*top:1715px;*/
 top:330px;
 width:660px;
 text-align:center;
 opacity:0;
}

 #caption4 {
 position: absolute;
 font-family: sans-serif;
 font-weight: bold;
 font-size:150%;
 text-shadow: black 1px 1px 3px;
 z-index:1;
 /*top:1715px;*/
 top:355px;
 width:660px;
 text-align:center;
 opacity:0;
}
```

Figura 121: Ejemplo de CSS para un subtitulo(2)

Al haber cuatro subtítulos diferentes, se ha tenido que hacer cuatro estilos diferentes para diferenciarlos a la hora de manipularlos correctamente.

5.4. Evaluación. Pruebas de acceso en distintos agentes de usuario

Después de la implementación de la página web con player en HTML5, queda lo más importante: comprobar que la página y player funcionan en los navegadores más conocidos y más usados. Se comprobará el nivel de soporte que ofrece HTML5 para sus diferentes etiquetas y elementos que se han presentado en el capítulo dos, y además de indicar las incidencias encontradas, se aportará una tabla cuya información será suministrada por una página web encargada de comprobar dicho nivel[9] y que mostrará una puntuación con respecto a este resultado.

Aprovechando estas pruebas, se ha probado el soporte que presentan estos navegadores con el nuevo estándar de hojas de estilos CSS3 [W3C, CSS3 2011]. Aunque no era objetivo del proyecto, al ser utilizada esta tecnología, se han dado incidencias a resolver y por ello que se haya podido hacer una pequeña aproximación del soporte ofrecido actualmente en los navegadores escogidos. Esta nueva versión de hojas de estilo, presenta nuevas características que aportan a los navegadores estilos más dinámicos y modernos, y formas de manipular la presentación de las páginas más fácil para el desarrollador.

Los navegadores seleccionados para las pruebas son los siguientes:

- Mozilla Firefox[10]: de sobra conocido por la mayoría de la gente, es de los navegadores más usados y más populares a nivel mundial, y famoso por ser de los más rápidos a la hora de navegar por la Web.
- Microsoft Internet Explorer[11]: quizá sea el navegador más conocido al ser incluido en el sistema operativo Windows, pero no necesariamente el mejor, en nuestro caso se verá su nivel de adaptación de HTML5.
- Chrome[12]: uno de los navegadores más recientemente lanzados por la empresa Google [Google, 2011].
- Opera[13]: quizá sea menos conocido por el resto, pero no menos importante ya que es uno de los que apoyó HTML5 casi desde que se anunció y su nivel de adaptación se verá en este capítulo.
- Safari[14]: es el navegador de Apple y más conocido por los usuarios del sistema Operativo Mac OS, aunque también está disponible una versión para Windows.

Un factor importante que hay que dar a conocer, es que las principales diferencias que hay entre los navegadores es la interpretación del código tanto HTML como CSS, por lo que para cada navegador se ha tenido que hacer hojas de estilo diferentes y el código HTML5 es exactamente igual para todos.

[9] www.html5test.com
[10] http://www.mozilla-europe.org/es/
[11] http://windows.microsoft.com/es-ES/internet-explorer/products/ie/home
[12] http://www.google.com/chrome?hl=es
[13] http://www.opera.com/
[14] http://www.apple.com/es/safari/

Se comentará en cada navegador el soporte que ofrece este nuevo estándar, por lo que se ha tenido que estudiar su nivel de adaptación para adaptar el código y los objetivos del proyecto para una presentación correcta evitando diferencias entre ellas.

También se comentará si ha habido alguna incidencia con algún navegador en particular que no ocurriese con otros, y más adelante las soluciones que se han tomado.

5.4.1. Mozilla Firefox

En el navegador Mozilla Firefox en su versión más reciente 4.0, el soporte que presenta para HTML5 es muy alto, aceptando un gran número de elementos y etiquetas presentes en este nuevo estándar. Las más importantes para este proyecto como <video> y <audio> no presentan ninguna dificultad en el momento de interpretar el código de estas etiquetas, incluyendo las no menos importantes <header>, <aside>, <section>, <article> y <footer>.

Las diferencias entre la nueva versión 4.0 y la anterior 3.6 son minúsculas, aunque adaptando lógicamente, nuevas funcionalidades que la anterior no tenía, pero que no afectan directamente a los objetivos presentados en el capítulo 1.

Aprovechando con la experimentación de este navegador, se ha comprobado también el soporte que ofrece de CSS3, probando nuevas funcionalidades como el redondeado de bordes, los colores degradados normalmente usados para fondos en la página y también funcionalidades relacionadas con el tratamiento de imágenes de fondo.

El navegador Firefox soporta muchos de los elementos de HTML5 como las cabeceras y el Doctype, canvas, vídeo y audio exceptuando las extensiones que soporta tal y como se vio en el apartado 2.6.1 y 2.6.2, algunos tipos definidos en la etiqueta <form> para formularios no están completamente soportados, y algunos conceptos de seguridad tampoco están aceptados.

En la siguiente tabla se verá un resumen más extenso:

	Si	No	Parcialmente
<!DOCTYPE html>	×		
HTML5 tokenizer	×		
Canvas	×		
Video	×		
Audio	×		
Elementos de sección	×		
Elementos de agrupación de contenido	×		
Elementos semánticos a nivel de texto			×
Elementos interactivos			×
Form (formularios)			×
Interacción con el usuario	×		
Microdata		×	
Aplicaciones Web	×		
Seguridad		×	

	Si	No	Parcialmente
Ubicación geográfica	×		
WebGL(contexto 3D)	×		
Comunicación			×
Archivos			
FileReader API	×		
FileWriter API		×	
Almacenamiento	×		
Dispositivos locales		×	
Puntuación basada en el análisis de www.html5test.com en Abril de 2011	255		

Tabla 11: Tabla de soporte de HTML5 para Mozilla Firefox versión 4.0

5.4.2. Microsoft Internet Explorer

Para el navegador Microsoft Internet Explorer en su versión 8, el nivel de adaptación de HTML5 que ofrece es muy bajo, sólo ofreciendo soporte a etiquetas relacionadas con la estructura principal como <header>, <article>, <aside> y <footer>, pero las relacionadas con los objetivos del proyecto <video> y <audio> no las soportan.

En cambio, la nueva versión 9 todavía en fase Beta, aumenta mucho el soporte de HTML5 acercándose mucho a lo ofrecido por Firefox. Aun así, habrá que esperar a comprobar la versión 9 definitiva de este navegador para comprobar su nivel adaptación final a este nuevo estándar, aunque viendo el nivel experimentado en su fase Beta, se espera que aumente y mejore significativamente.

En cuanto al soporte de CSS3 en la versión 9 Beta, es muy similar a Firefox pero aún no acepta algunas nuevas características como la manipulación de fondos o colores degradados.

Las incidencias que se han ido encontrando a la hora de programar la página en HTML5, son pocas, ya que como se ha mencionado anteriormente, el código HTML5 es exactamente el mismo para todos. En cambio, a la hora de programar en JavaScript y CSS sí se han encontrado varios inconvenientes.

- Algunas líneas de código JavaScript, han dado problemas por incompatibilidad de algunas instrucciones. Esto se debe a que se suele dar el caso que algunas instrucciones de este lenguaje orientado a objetos suele ser ligeramente diferentes para cada navegador, ya que lo que funcionaba perfectamente en Firefox, no lo hacía en Internet Explorer debido a que directamente informaba de error en la consola de JavaScript. Por lo tanto se tuvo que estudiar las diferencias que existían en estas instrucciones y realizar una parte de código expresamente dirigida a cuando se usara IE como navegador.

Esta instrucción conflictiva era “XMLHttpRequest”, la cual creaba un objeto para poder acceder al archivo XML que contenía los subtítulos que posteriormente se iban a exponer en el video. El nombre que se le pasaba por parámetro era diferente para IE, y además no permitía el acceso al archivo en modo local(alojado en el ordenador), sino que tenía que estar en un servidor, por lo que no se pudo comprobar que estas instrucciones funcionaban correctamente hasta que la página no estuvo colgada en la red.

- En CSS se tuvo que adaptar una hoja de estilo propio para IE, ya que la interpretación que se hacía de la de Firefox no era la misma, por lo que se tuvo que revisar y comprobar qué instrucciones aceptaba y cuáles no.

En la siguiente tabla se mostrará un análisis más extenso:

	Si	No	Parcialmente
<!DOCTYPE html>	×		
HTML5 tokenizer		×	
Canvas	×		
Video	×		
Audio	×		
Elementos de sección	×		
Elementos de agrupación de contenido	×		
Elementos semánticos a nivel de texto			×
Elementos interactivos			×
Form (formularios)			×
Interacción con el usuario		×	
Microdata		×	
Aplicaciones Web		×	
Seguridad		×	
Ubicación geográfica	×		
WebGL(contexto 3D)		×	
Comunicación			×
Archivos			
FileReader API		×	
FileWriter API		×	
Almacenamiento			×
Dispositivos locales		×	
Puntuación basada en el análisis de www.html5test.com en Abril de 2011	130		

Tabla 12: Tabla de soporte de HTML5 para Internet Explorer 9 Beta

5.4.3. Chrome

Con el navegador Chrome en su versión 10, su nivel de soporte de HTML5 es muy alto. Este navegador es uno de los más recientes en los últimos años que han salido disponibles para su uso gratuiro, y junto con Firefox acepta un gran número de etiquetas y elementos incluido <video> y <audio>, e incluso superando a Firefox. En la página mencionada anteriormente obtiene la puntuación más alta de los navegadores con los que se han hecho estas pruebas, mostrando una gran adaptación a este nuevo estándar, tal y como se puede ver en la tabla 13 de resumen para Chrome.

El único inconveniente que se ha tenido desde el principio con Chrome, es la imposibilidad de que la función "XMLHttpRequest" funcionase de forma local, teniendo que esperar hasta que la página estuviese colgada en la red, al igual que pasó con IE 9.

En cuanto a CSS, no ha habido mayores problemas que los comentados para Internet Explorer de instrucciones que no eran igualmente interpretadas que el resto de navegadores, y de crear una hoja de estilo específica para Chrome.

En la siguiente tabla se muestra un resumen más extendido:

	Si	No	Parcialmente
<!DOCTYPE html>	×		
HTML5 tokenizer	×		
Canvas	×		
Video	×		
Audio	×		
Elementos de sección	×		
Elementos de agrupación de contenido	×		
Elementos semánticos a nivel de texto			×
Elementos interactivos			×
Form (formularios)			×
Interacción con el usuario	×		
Microdata		×	
Aplicaciones Web	×		
Seguridad			×
Ubicación geográfica	×		
WebGL(contexto 3D)	×		
Comunicación	×		
Archivos			
FileReader API		×	
FileWriter API		×	
Almacenamiento			×
Dispositivos locales		×	
Puntuación basada en el análisis de www.html5test.com en Abril de 2011	288		

Tabla 13: Tabla de soporte de HTML5 para Chrome 10

5.4.4. Opera

En el navegador Opera el soporte de HTML5 es muy buena, teniendo en cuenta que este navegador fue uno de los que apoyó este nuevo estándar y uno de los primeros en adaptarse.

En cambio, teniendo en cuenta esto, su nivel de soporte podría haber sido bastante mayor ya que es un navegador que ya lleva bastantes años disponible para los usuarios. Sin embargo, a pesar de su alta puntuación en la página donde se ha obtenido la información para

este análisis, incomprensiblemente no soporta las etiquetas más importantes de HTML5 <header>, <aside>, <section>, <article> y <footer> mientras que <video> y <audio> sí los soporta, y aunque no lo soporte, el navegador es capaz de mostrar la página igual que el resto de navegadores probados, con la excepción de que el elemento <aside> no lo sitúa donde debería situarlo.

En cuanto a CSS, no ha habido mayores problemas que los mencionados anteriormente en otros navegadores, teniendo que hacer una hoja de estilo propia para Opera y adaptar las instrucciones a una que sea reconocida por el navegador.

Sin embargo, ha habido problemas a la hora de optimizar el código JavaScript. Como se ha mencionado anteriormente en Chrome y IE 9 Beta, ha habido algunos incidentes con la instrucción "XMLHttpRequest", la que después de una actualización de Opera, impedía ejecutar esta instrucción de forma local, es decir, teniendo los ficheros en el ordenador, por lo que no se pudo comprobar que funcionaba hasta que no estuvo colgada en la Web.

En la siguiente tabla resumen, se verá el nivel de soporte en Opera:

	Si	No	Parcialmente
<!DOCTYPE html>	×		
HTML5 tokenizer		×	
Canvas	×		
Video	×		
Audio	×		
Elementos de sección		×	
Elementos de agrupación de contenido		×	
Elementos semánticos a nivel de texto			×
Elementos interactivos			×
Form (formularios)			×
Interacción con el usuario		×	
Microdata		×	
Aplicaciones Web			×
Seguridad		×	
Ubicación geográfica	×		
WebGL(contexto 3D)		×	
Comunicación			×
Archivos			
FileReader API		×	
FileWriter API		×	
Almacenamiento			×
Dispositivos locales		×	
Puntuación basada en el análisis de www.html5test.com en Abril de 2011	214		

Tabla 14: Tabla de soporte de HTML5 para Opera

5.4.5. Safari

El navegador Safari da un buen soporte de HTML5, siendo este navegador el más conocido entre los usuarios de Mac OS al ser la conocida empresa Apple su responsable. En la página mencionada anteriormente para obtener el análisis del navegador, obtiene una buena puntuación con respecto al resto de navegadores, ya que da bastante soporte a todas las funcionalidades que ofrece HTML5 como <video> y <audio>, la mayoría de elementos y bastantes tipos de formularios.

En cuanto a CSS, comentar que no ha dado mayores problemas que el resto de navegadores, teniendo que hacer una hoja de estilo propia para este navegador y adaptar algunas instrucciones específicas.

En JavaScript no ha dado ninguna incidencia con respecto al resto de navegadores, mostrando una gran flexibilidad a la hora de programar y teniendo en cuenta que proviene de un S.O. no muy extendido, pero que con el tiempo se va aceptando más entre los usuarios asiduos a la informática.

En la siguiente tabla resumen se puede ver el nivel de adaptación a HTML5.

	Si	No	Parcialmente
<!DOCTYPE html>	×		
HTML5 tokenizer		×	
Canvas	×		
Video	×		
Audio	×		
Elementos de sección	×		
Elementos de agrupación de contenido		×	
Elementos semánticos a nivel de texto			×
Elementos interactivos			×
Form (formularios)			×
Interacción con el usuario	×		
Microdata		×	
Aplicaciones Web	×		
Seguridad			×
Ubicación geográfica	×		
WebGL(contexto 3D)		×	
Comunicación	×		
Archivos			
FileReader API		×	
FileWriter API		×	
Almacenamiento			×
Dispositivos locales		×	
Puntuación basada en el análisis de www.html5test.com en Abril de 2011	228		

Tabla 15: Tabla de soporte de HTML5 para Safari

5.4.6. Discusión de los datos

La conclusión que podemos sacar de las pruebas realizadas en distintos navegadores, es que todavía falta mucho para que el nuevo estándar HTML5 sea soportado de manera completa con todas sus funcionalidades y se asiente totalmente como el "único" lenguaje para programar HTML en la Web.

Muchos de los nuevos elementos más significativos son soportados por los navegadores, pero aún queda por adaptar casi todos los tipos de formularios ya que en este nuevo estándar se puede definir el tipo de datos que se va a introducir en cuadro de texto correspondiente a un formulario, y actualmente se da soporte a muy pocos en relación a todas las posibilidades que ofrece.

En cuanto a sintaxis de HTML5 hay de todo: aún hay navegadores que no soportan las nuevas etiquetas, y que parece imprescindible para dar un buen soporte al nuevo estándar, pero en cambio las etiquetas <video> y <audio> están aceptadas para todos los navegadores si exceptuamos el problema de codecs que existe en la actualidad, ya que cada navegador acepta los códec que la empresa responsable considere, por lo que en este momento hay un pequeño caos hasta que no se imponga uno para todos.

Para las nuevas funcionalidades como geolocalización y demás novedades, hay diversas opciones ya que cada navegador da soporte a unas y a otras no, por lo que es cuestión de tiempo que los desarrolladores lo tomen en cuenta para incluirlo en nuevas versiones.

5.4.7. Incidencias

A lo largo de la realización de la página interfaz con el player, se han sucedido numerosas incidencias relacionadas en su mayoría con problemas de código.

El que más tiempo ha ocupado ha sido la instrucción "XMLHttpRequest". Esta instrucción crea un objeto dado, un archivo XML para poder manipularlo mediante el DOM del navegador. El problema era que algunos navegadores no permitían ejecutar esta instrucción de forma local, teniendo que estar el archivo al que se quería acceder en la red, por lo que a la hora de programar en local no sabía si iba a funcionar o no. El motivo por el que tuve que usar esta instrucción y no otra, es que la que usaba al principio no lo soportaban algunos navegadores, y directamente lanzaban un error en la consola de JavaScript.

Otro inconveniente que tuve que subsanar, fue mi bajo nivel de conocimiento de JavaScript entre otras tecnologías usadas en el proyecto.

Capítulo 6
6. Presupuesto

En este capítulo se listan los costes asociados a los medios materiales y a los recursos humanos utilizados en el desarrollo de este proyecto.

Se ha planificado un calendario laboral que implica una jornada de cinco horas diarias. Aunque la línea de realización del proyecto ha sido irregular a lo largo de todo el proceso, se ha supuesto una dedicación continua. Por lo que la duración del proyecto y el número de jornadas refleja la media real de las horas invertidas en el desarrollo del proyecto al completo.

La distribución de los trabajos realizados se ha divido en las siguientes fases.

- Fase de análisis y estudio: en esta fase se realizará un análisis de las capacidades que ofrece el nuevo estándar HTML5, y un estudio de las posibilidades de desarrollar una página o interfaz que cubra las necesidades u objetivos previamente definidos en el análisis.
- Fase de diseño: esta fase incluye el proceso de diseño de la página web que reproduzca contenido multimedia video para implementar en HTML5.
- Fase de desarrollo y pruebas: esta última fase se compone del desarrollo de la página web diseñada en la fase anterior mediante las técnicas de programación adecuadas, y la realización de pruebas.

En los costes de equipo se han tenido en cuenta solo el material utilizado para realizar el proyecto tales como ordenadores y portátiles, dado que el servidor donde se alojará la página o herramienta lo aporta la Universidad.

Según las cifras establecidas en la hoja de cálculo del desglose presupuestario del proyecto, se puede determinar que el presupuesto total de este proyecto asciende a la cantidad de trece mil novecientos setenta euros (13.970 €).

1.- Autor:

Alberto Sánchez-Heredero Pérez

2.- Departamento:

Informática

3.- Descripción del Proyecto:

- Titulo	**Accesibilidad a los contenidos audiovisuales en la Web a través de HTML5**
- Duración (meses)	**4,2**
Tasa de costes Indirectos:	**20%**

4.- Presupuesto total del Proyecto (valores en Euros):

Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
		Jefe del proyecto	0,5	5.600,00	2.800,00	
		Programador	2,2	2.200,00	4.840,00	
		Diseñador	1,5	2.600,00	3.900,00	
		Hombres mes	**4,2**	**Total**	11.540,00	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
PC Intlel Core™ i3 550 4GB RAM 500GB	499,00	100	5	60	41,58
Monitor LCD LED 2011x 50,8 cm	119,00	100	5	60	9,92
Portátil Intel® Core™ i3-350M 4GB RAM 640GB	599,00	100	5	60	49,92
				Total	101,42

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} xCxD$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo:

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	11.540
Amortización	101
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	2.328
Total	13.970

Capítulo 7
7. Conclusiones y líneas futuras

La motivación que me ha llevado a elegir este tema como mi proyecto de fin de carrera, son mis ganas por conocer más a fondo la programación web ya que desde hace tiempo me ha interesado y gustado mucho, y siempre me hubiese gustado conocer la forma en la que se programan las páginas web que casi toda mi vida he visitado, y este proyecto me ha dado la oportunidad de conocerlo desde otro punto de vista.

Me parece un tema muy interesante porque así no solo aprendo como estudiante a hacer cosas nuevas, sino que también aprendo a ayudar a otro tipo de personas que necesitan la atención de lo que para el resto de las personas no les supone ninguna dificultad añadida, y eso hace que tu trabajo sea recompensado sabiendo que ha servido para ayudar a muchas personas con discapacidad y que tienen el mismo derecho que los demás al acceso a la información en Internet.

Las conclusiones que se pueden obtener de este proyecto de fin de carrera, es que se han cumplido los objetivos propuestos al inicio del documento.

- Estudio de la cuestión de:
    - HTML5: se ha llevado un estudio del último borrador disponible para su lectura, analizando lo nuevo que aporta y futuras ampliaciones, y una comparación con la versión anterior 4.01.
    - Accesibilidad web: se ha llevado a cabo un estudio de la accesibilidad en la web, de todas las normas y estándares proporcionadas por el W3C, en la que se aportan una serie de recomendaciones para hacer una web accesible para todos. Las normas estudiadas han sido WCAG (Pautas de accesibilidad al contenido en la Web), ATAG (Pautas de accesibilidad para Herramientas de Autor), WAI-Aria (*Web Accesibility Initiative-Accesible Rich Internet Application*), y la más importante para este proyecto las UAAG(Pautas de accesibilidad para Agentes de Usuario).
    - Accesibilidad de los contenidos audiovisuales en la Web: para este estudio se ha obtenido información del libro “Accesibilidad a los contenidos audiovisuales en la web: Una panorámica sobre legislación, tecnologías y estándares” proporcionado por Lourdes Moreno, en el que se muestra cómo ofrecer accesibilidad a los contenidos audiovisuales en la Web, siguiendo las recomendaciones de WCAG, mencionada en el punto anterior.

- Estudio de la especificación HTML5: para este objetivo se ha llevado a cabo un exhaustivo estudio de las especificaciones técnicas propuestas en HTML5. Se realizó un estudio de los nuevos elementos y etiquetas incluidas en el estándar, ofreciendo un análisis de estos elementos y etiquetas y su definición en HTML5.
- Contenido multimedia accesible con HTML5: para este apartado se ha realizado un análisis más detallado de lo explicado en la especificación de HTML5, separando los elementos que realmente mejoran la accesibilidad tanto a nivel de desarrollador como a nivel de usuario.
- Caso de prueba:
    - Desarrollo de una interfaz con reproductor con HTML5: para comprobar la información expuesta en este documento, se ha realizado un página web en HTML5 con el fin de poner en práctica los conocimientos obtenidos y aplicar las normas y estándares estudiados, y con el objetivo en concreto de aportar contenido multimedia accesible utilizando este nuevo estándar y otros lenguajes necesarios, como ha sido el caso de JavaScript.
    - Diseño de la interfaz de acuerdo a los objetivos planteados y requisitos de accesibilidad a cumplir.
    - Implementación de los conocimientos adquiridos mediante el estudio del nuevo estándar HTML5 y de los estándares: WCAG y UAAG en el desarrollo de la página web.
    - Testeo en distintos navegadores: una vez realizado el desarrollo de una interfaz en HTML5, se ha tenido que probar en distintos navegadores para valorar el nivel de implementación.
    - Evaluación de los resultados obtenidos, tanto de la página web implementada en HTML5 como del testeo en distintos navegadores.
    - Discusión de los datos: con los resultados obtenidos, se ha hecho una valoración de los navegadores y el soporte que ofrecen de HTML5.

Una vez terminado el documento y comprobado que se han cumplido los objetivos propuestos, podemos obtener unas conclusiones.

HTML5 es un gran salto en cuanto a programación web se refiere, actualizando uno de los lenguajes más conocidos y más importantes en la que está escrita la Web. Introduce muchas mejoras que la anterior versión no tenía, ya que era una versión muy longeva, añadiendo nuevas características que ya eran necesarias, como una forma de estructurar el código más clara y fácil, nuevas etiquetas como <video> y <audio> que prometen una revolución ya que proponen la universalidad de estos contenidos multimedia desde el punto de vista del programador.

Aunque esta universalidad, trae otros problemas derivados de los contenidos multimedia, y estos son los códecs. El problema radica en el tipo de formato que cada navegador soporta, ya que detrás hay una empresa con sus propios intereses, por lo que actualmente es difícil encontrar una solución que satisfaga a todas las empresas involucradas tanto de códecs como de navegadores.

Por otro lado, se espera que HTML5 sea soportado por la gran mayoría de navegadores presentes en los ordenadores, pero otro mercado en auge también debería tenerse en cuenta, y es el mercado de los dispositivos móviles, ya sean los sistemas operativos iOS o Android. Si HTML5 no se adapta a este mercado, sería una gran pérdida para su universalización, ya que los dispositivos móviles crecen año tras año y se debería tener en consideración.

También se debe comentar sobre las normas y estándares recogidas en este documento, y es que HTML5 no da un soporte total de WCAG y UAAG. En este proyecto, precisamente se ha buscado dar soporte a estas recomendaciones por medios indirectos a HTML5, es decir, por otros lenguajes complementarios como JavaScript para ofrecer subtitulado y audiodescripción, lo que HTML5 no ofrece por sí solo.

Para el desarrollo de estas funcionalidades, se han tenido en cuenta diversas páginas de desarrollo, como "*A more accesibility <video> player*" [Dev. Opera, 2011 a], "*Terrill Thompson: Creating your own Accesibility HTML5 Media Player*" [Therrill Thompson Blogspot, 2009], "Introduction to HTML5 Video" [Dev. Opera, 2011 c], y la base que fue la idea para usar JavaScript "*Accesible HTML5 vídeo with JavaScript captions*" [Dev. Opera, 2011 b].

Es muy posible que en un periodo de corto a medio plazo, estas carencias se irán subsanando y además se incluirán otras que complementarán a las ya mencionadas, ya que HTML5 no va a ser un lenguaje estático sino que con el tiempo irá mejorando y añadiendo nuevas funcionalidades y el resto de navegadores le darán soporte totalmente, ya que este nuevo estándar será el futuro de la programación en la web.

Durante el desarrollo de este proyecto de fin de carrera, debido al factor tiempo, no se han podido implementar algunas funcionalidades que hubiesen mejorado sustancialmente la página web desarrollada para este fin, y que hubiesen cumplido algunos de los puntos de las UAAG 2.0. Por lo tanto se van a nombrar una serie de características se hubiesen podido incluir pero que quedan como posibles ampliaciones futuras:

- Posibilidad de visualizar el vídeo en pantalla completa manteniendo los controles creados en HTML5.
- Botón de reducción y aumento de tamaño de letra de los subtítulos.
- Barra de carga del contenido del video.
- Recordatorio de las opciones del usuario entre diferentes sesiones relacionado con la elección del idioma de los subtítulos o la habilitación de audiodescripción.

Glosario

- **Agentes de usuario**: Navegadores, reproductores multimedia, tecnología de asistencia y otro software que usa la gente para acceder e interactuar con el contenido web.
- **Audiodescripción**: La narración agregada a la pista de sonido para describir los detalles visuales importantes que no se pueden entender sólo con la banda de sonido principal. Notas:
    - La audiodescripción del vídeo proporciona información sobre las acciones, personajes, cambios de escena, textos que aparecen en pantalla y otros contenidos visuales.
    - En las audiodescripciones estándares, la narración se añade durante las pausas existentes en el diálogo. (Véase también audiodescripción ampliada.)
    - Cuando toda la información sobre el vídeo ya se proporciona en el audio de la presentación, no es necesaria ninguna audiodescripción adicional.
    - En inglés también se la denomina "video description" (descripción de vídeo) o "descriptive narration" (narración descriptiva).
- **Contenedor multimedia**: Un formato contenedor es un tipo de formato de archivo que almacena información de vídeo, audio, subtítulos, capítulos, meta-datos e información de sincronización siguiendo un formato preestablecido en su especificación.
- **CSS**: acrónimo de **C**ascading **S**tyle **S**heets(Hojas de estilo en cascada) es un mecanismo simple basado en reglas, es decir, declaraciones sobre el estilo de uno o más elementos, el cual describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información en un documento. Se utiliza para dar estilo a documentos HTML y XHTML, separando el contenido de la presentación
- **HTML**: acrónimo de **H**yper **T**ext **M**arkup **L**anguage, es el lenguaje de marcado predominante para la elaboración páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.
- **JavaScript**: es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador, permitiendo mejoras en su interfaz de usuario y páginas web dinámica.
- **Página web**: es un documento o información electrónica adaptada para la World Wide Web que generalmente forma parte de un sitio web. Su principal característica son los hipervínculos de una página, siendo esto el fundamento de la WWW.
- **Productos de apoyo, Ayudas técnicas**: (igual que Tecnología de apoyo (UNE EN ISO 9999), en inglés Assistive technologies): Software y hardware que utiliza la gente con discapacidad para mejorar sus posibilidades de interacción con la Web. Algunos ejemplos de este tipo de tecnología son los lectores de pantalla o el software que facilita la entrada por voz. Una definición un poco más formal de este tipo de software es cualquier entidad, parte, producto, sistema o software que adquirido comercialmente, modificado o personalizado, que se utiliza para incrementar, mantener o mejorar las capacidades funcionales de un individuo con discapacidad. Son dispositivos o programas que actúan como agentes de usuario, o conjuntamente con un agente de

usuario de amplia distribución, para proporcionar una funcionalidad que cumpla con los requisitos de los usuarios con discapacidades que están más allá de la funcionalidad proporcionada por el agente de usuario de amplia distribución.

- **Plug-in**: traducido como complemento, es una aplicación que se relaciona con otra para aportarle una nueva función o característica. Frecuentemente se le relaciona con complementos de navegadores web, con el fin de ampliar sus funciones, siendo posiblemente el más conocido el plug-in de Flash para ver contenido multimedia tales como video.
- **Subtitulad**o (entendido como subtítulos para sordos): Alternativas visuales y/o alternativas textuales, sincronizadas, tanto para la información sonora hablada como no hablada, necesarias para comprender el contenido multimedia. Notas:
    - Los subtítulos para sordos son similares a los subtítulos que presentan sólo el diálogo, excepto por que los subtítulos para sordos transmiten no sólo el contenido del diálogo sino también equivalentes para la información sonora que no es diálogo y que es necesaria para comprender el contenido del programa, incluyendo efectos sonoros, música, risas, indentificación del hablante y localización.
    - En algunos países de lengua inglesa, se le llama "subtitles" también a lo que en otros se llama "caption". Nota de la traducción: En el Reino Unido se le llama subtítulos tanto a los subtítulos en una lengua distinta a la original de la pista de diálogos, como a los subtítulos para sordos. En Estados Unidos, Canadá, Nueva Zelanda y Australia se hace la diferenciación utilizando el término "captions" para referirse a los subtítulos para sordos.
- **Subtítulos:** Texto alternativo o visual sincronizado con la locución y el audio de un contenido multimedia necesario para comprender el contenido del mismo.
- **World Wide Web**: es un sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet. Con un navegador web, un usuario visualiza sitios web compuestos de páginas web que pueden contener texto, imágenes, videos u otros contenidos multimedia, y navega a través de ellas usando hiperenlaces.

Referencias

[AENOR, 2004] AENOR, Asociación Española de normalización y certificación, Norma UNE 139803:2004: Requisitos de accesibilidad para contenidos en la Web www.aenor.es,2004. http://www.inteco.es/Accesibilidad/Normativa_1/Descarga/DescargaUNE_139803

[AENOR, 2003] AENOR, Asociación Española de normalización y certificación. Subtitulado para personas sordas y personas con discapacidad auditiva. Subtitulado a través del teletexto. UNE 153010:2003. http://www.aenor.es

[AENOR, 2005] AENOR, Asociación Española de normalización y certificación, Audiodescripción para personas con discapacidad visual. Requisitos para la audiodescripción y elaboración de audioguías, UNE 153020:2005, http://www.aenor.es

[AENOR, 2009]_Asociación Española de Normalización y Certificación, 2009, UNE 139802:2009: Requisitos de accesibilidad del software, http://www.aenor.es

[Adobe, 2008] Adobe Premiere, http://www.adobe.com/es/products/premiere/

[Adobe, 2009] http://www.adobe.com/es/products/flashplayer/

[Adobe, 2011] Adobe Flash FLA to HTML http://labs.adobe.com/technologies/wallaby/

[Apple, 2011 a] Apple http://www.apple.com/es/

[Apple, 2011 b] Apple, Reproductor quicktime, http://www.apple.com/quicktime/

[BOE, 2002] Boletín Oficial del Estado, LEY 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico. 2002. Disponible en: http://www.boe.es/boe/dias/2002/07/12/pdfs/A25388-25403.pdf

[BOE, 2003] Boletín Oficial del Estado, LEY 51/2003, de 2 de diciembre, de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad. 2003. http://www.boe.es/boe/dias/2003/12/03/pdfs/A43187-43195.pdf

[BOE, 2008] INSTRUMENTO de Ratificación de la Convención sobre los derechos de las personas con discapacidad, hecho en Nueva York el 13 de diciembre de 2006. 21 de abril de 2008. http://www.boe.es/boe/dias/2008/04/21/pdfs/A20648-20659.pdf

[Clark, J., 2004] Best practices in online captioning,

http://joeclark.org/access/captioning/bpoc/

[Dev. Opera, 2011 a] http://dev.opera.com/articles/view/more-accessible-html5-video-player/

[Dev. Opera, 2011 b] http://dev.opera.com/articles/view/accessible-html5-video-with-javascripted-captions/

[Dev. Opera, 2011 c] http://dev.opera.com/articles/view/introduction-html5-video/

[EU, 2002] Informe Final eEurope, 2002. http://eur-lex.europa.eu/LexUriServ/site/es/oj/2003/c_220/c_22020030916es00360038.pdf

[González M. et al, 2011]. María González-García, Lourdes Moreno, Paloma Martínez, Ana Iglesias, (2011). Requisitos de accesibilidad web en los reproductores multimedia, Interacción 2011. XII

Congreso de Interacción Persona-Ordenador, Lisboa, Portugal, September, 2011. http://interaccion2011.aipo.es/

[Google, 2011] www.google.com

[Henry S., 2007] Henry, Shawn Lawton. Just Ask: Integrating Accessibility Throughout Design. Madison, WI: ET\Lawton, www.uiAccess.com/justask/

[Hi Software, 2011] Hi-Caption Studio, http://hisoftware.com/

[IBM, 2005] CaptionMeNow, http://www.ibm.com

[INTECO, 2010] Guía de WAI-ARIA http://www.inteco.es/Accesibilidad/Formacion_6/Manuales_y_Guias/WAI_ARIA

[ISO, 2008 b] International Standards for Business, Government and Society (ISO). 9241-171:2008. Ergonomics of human-system interaction -- Part 171: Guidance on software accessibility. 2008. Disponible en: http://www.iso.org/

[ISO, 2008 c] International Standards for Business, Government and Society (ISO). ISO 9241-20:2008, Ergonomics of human-system interaction -- Part 20: Accessibility guidelines or information/communication technology (ICT) equipment and servicesISO 9241-20, 2008. Disponible en: http://www.iso.org/

[ISO, 2008 d] International Standards for Business, Government and Society (ISO). ISO 9241-151:2008, Ergonomics of human-system interaction -- Part 151: Guidance on World Wide Web user interfaces, 2008. Disponible en: http://www.iso.org/

[Leanback Player, 2011] Leanback Player HTML5 Video Player http://dev.mennerich.name/showroom/html5_video/

[Microsoft, 2003] Microsoft Synchronized Accessible Media Interchange (SAMI),, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnacc/html/atg_samiarticle.asp

[Microsoft, 2008] Windows Media, http://www.microsoft.com/windows/windowsmedia/player/default.aspx

[Moreno L. et al, 2006] L. Moreno, A. Iglesias, J.M. Carrero y P. Martínez. Subtitulado y Audiodescripción en páginas Web accesibles, I Congreso de Accesibilidad a los medios audiovisuales para personas con discapacidad, AMADIS´06, Madrid España, Julio 2006. Artículo [PDF] / Presentación [PDF]

[Moreno L. et al., 2008 a] Application of disability standards for including multimedia on the Web. Moreno L., Martínez P. and Ruiz B. IEEE MultiMedia Special Issue on Accessibility, October-December 2008.

[Moreno, L. et al., 2008 b] Lourdes Moreno, Belén Ruiz-Mezcua, Paloma Martínez, Juan Manuel Carrero, Juan Ramón Martínez, (2008). Accesibilidad a los contenidos audiovisuales en la Web: Una panorámica sobre legislación, tecnologías y estándares (WCAG 1.0 y WCAG 2.0), November, 2008, Editorial del Real Patronato sobre Discapacidad, ISBN: 978-84-691-77. http://www.cesya.es/files/documentos/accesibilidad_contenidos.pdf

[Moreno L., 2010] Lourdes Moreno. Tesis Doctoral. AWA, Marco metodológico específico en el dominio de la accesibilidad para el desarrollo de aplicaciones web http://labda.inf.uc3m.es/doku.php?id=es:labda_asignacion_tesis

[NCAM, 2003] Media Access Generator (MAGpie), http://ncam.wgbh.org/webaccess/magpie/

[NCAM, 2006] NCAM, Accessible Digital Media. Design Guidelines for Electronic Publications, Multimedia and the Web, 2006, http://ncam.wgbh.org/publications/adm/disabilities.html

[Realnetworks, 2008] Reproductor RealPlayer, http://www.realnetworks.com/products/media_players.html

[Shneiderman, B, 2000] B. Shneiderman, ‘‘Universal Usability’’ Comm. ACM, vol. 43, no. 5, 2000, pp. 84-91, http://doi.acm.org/10.1145/332833.332843

[Therrill Thompson Blogspot, 2009] http://terrillthompson.blogspot.com/2010/08/creating-your-own-accessible-html5.html

[UN, 1948] Naciones Unidas – Declaración Universal de los Derechos Humanos, 1848. http://www.un.org/spanish/Depts/dpi/boletin/humanrights/universaldec.html

[UN, 1993] Naciones Unidas - Normas Uniformes sobre la igualdad de oportunidades para las personas con discapacidad, 1993. http://www.un.org/esa/socdev/enable/dissres0.htm

[W3C, 1998] HTML4.0 Specification http://www.w3.org/TR/1998/REC-html40-19980424/

[W3C, 1999 a] HTML4.01 Specification http://www.w3.org/TR/html401/

[W3C, 1999 b] W3C, WAI, Web Content Accessibility Guidelines 1.0, WCAG 1.0, 1999. http://www.w3.org/TR/WCAG10/

[W3C ATAG, 2005] Introducción a la accesibilidad Web http://www.w3c.es/traducciones/es/wai/intro/atag

[W3C UAAG, 2005] Introducción a UAAG http://www.w3c.es/traducciones/es/wai/intro/uaag

[W3C, 2006 a] Timed Text (TT), http://www.w3.org/AudioVideo/TT/

[W3C, 2008 a] W3C, Cascading Style Sheets Home Page http://www.w3.org/Style/CSS/

[W3C, 2008 b] W3C, Pautas de Accesibilidad al Contenido en la Web 2.0, Web Content Accessibility Guidelines 2.0, http://www.w3.org/TR/WCAG20/

[W3C, 2008 c] W3C, Synchronized Multimedia Integration Language (SMIL 3.0), http://www.w3.org/TR/2008/PR-SMIL3-20081006/

[W3C, 2008 d] W3C, Web Content Accessibility Guidelines (WCAG) Overview , Editor: Shawn Lawton Henry., 2008. http://www.w3.org/WAI/intro/wcag.php

[W3C, 2010 a] Guía Breve de Web Semántica http://www.w3c.es/divulgacion/guiasbreves/websemantica

[W3C, 2010 b] User Agent Accesibility Guidelines 2.0 Requirements http://www.w3.org/TR/2007/WD-UAAG20-requirements-20071031/

[W3C, 2010 c] Web Accessibility Initiative (WAI) http://www.w3.org/WAI/

[W3C, 2011 a] HTML5 http://www.w3.org/TR/html5/

[W3C, 2011 c] Sobre el W3C – W3C España http://www.w3c.es/Consorcio/

[W3C, 2011 d] Accesible Rich Internet Applications (WAI-ARIA) 1.0 http://www.w3.org/TR/wai-aria/

[W3C, CSS3 2011] http://www.w3.org/TR/2011/WD-css3-text-20110412/

[W3Schools, 2010 a] HTML5 Audio http://www.w3schools.com/html5/html5_audio.asp

[W3Schools, 2010 b] HTML5 Event Attributes http://www.w3schools.com/html5/html5_ref_eventattributes.asp

[W3Schools, 2010 c] HTML5 Input Types http://www.w3schools.com/html5/html5_form_input_types.asp

[W3Schools, 2010 d] HTML5 Tag Reference http://www.w3schools.com/html5/html5_reference.asp

[W3Schools, 2010 e] HTML5 Video http://www.w3schools.com/html5/html5_video.asp

[Webaim, 2006] Creating Accessible Macromedia Flash Content, http://www.webaim.org/techniques/flash/

Anexo

Nivel de conformidad según las UAAG 2.0.

En este apartado se ha llevado a cabo una evaluación para analizar el nivel de cumplimiento de las pautas UAAG 2.0 [W3C, 2010 b] en el reproductor multimedia o *media player* desarrollado. En la tabla 16 se indica el *checklist* o tabla de puntos de revisión de las pautas. Esta evaluación ha sido realizada por un experto en el estándar UAAG para reproductores.

Guidelines	PFC	Observaciones
3.1.1 Identify Presence of Alternative Content(A)	✓	
3.1.2 **Configurable Default Rendering(A)**	×	
3.1.3 Browse and Render(A)	✓*	Puede intercambiar entre subtítulos y audio descripción
3.1.4 Rendering Alternative (Enhanced)(AA)	×	
3.5.1 **Highlighted items(A)**	✓	
3.6.1 Configure Text(A)	×	
3.7.1 Global Volume(A)	✓	
3.7.2 Speech Volume(A)	✓	
3.8.1 Speech Rate and Volume(A)	×	El volumen de la voz sí.
3.8.2 Speech Pitch and Range(AA)	×	
3.8.3 Advanced Speech Characteristics(AAA)	×	
3.8.4 Speech Features(AA)	×	
3.10.4 Resizable(A)	×	
3.11.3 User Interface Focus(A)	✓	
3.11.4 Extensions Focusable(A)	✓	
3.11.6 Retrieve Focus(A)	×	No hay ventanas anidadas
3.11.7 Return Focus(A)	✓	
3.11.8 Bi-Directional(A)	✓	
3.11.9 Sequential Navigation(A)	✓	
3.11.10 Only on User Request(A)	✓	
3.11.11 On Focus(A)	✓	
3.12.2 Outline View(AA)	×	
3.12.3 Configure Set of Important Elements(AAA)	×	
4.1.1 Keyboard Operation(A)	✓	

Guidelines	PFC	Observaciones
4.1.3 No Keyboard Trap (Minimum)(A)	✓	
4.1.4 Separate Selection from Activation(A)	✓	
4.1.6 Present Direct Commands in Rendered Content(A)	×	
4.1.7 Present Direct Commands in User Interface(AA)	✓	Menú con atajos de teclado
4.1.8 Keyboard Navigation(AA)	✓*	No es un grupo propiamente dicho, pero si puedes ir hacia delante y hacia atrás en el volumen
4.1.9 Important Command Functions(AA)	✓	
4.1.10 Override of UI Keyboard Commands(AA)	×	
4.1.11 User Override of Accesskeys(AA)	×	
4.1.12 Specify preferred keystrokes(AA)	×	
4.5.1 Change Preference Settings(A)	✓	
4.5.2 Persistent Accessibility Settings(A)	×	No se mantiene ni el subtítulo ni la audiodescripción en una nueva sesión.
4.5.3 Multiple Sets of Preference Settings(AA)	×	
4.5.4 Portable Preference Settings(AAA)	×	
4.5.5 Preferences Wizard(AAA)	×	
4.5.6 Restore all to default(A)	×	
4.5.7 Restore related preferences to default(AA)	×	
4.5.8 Change preference setting outside the UI(AA)	×	
4.6.1 Find(A)	×	
4.6.2 Find Direction(A)	×	
4.6.3 Match Found(A)	×	
4.6.4 Alert on No Match(A)	×	
4.6.5 Advanced Find(AA)	×	
4.7.5 Direct activation(AA)	✓	

Guidelines	PFC	Observaciones
4.7.6 Configure Set of Important Elements(AAA)	×	
4.7.7 Discover navigation and activation keystrokes(A)	×	
4.8.1 Configure Position(AAA)	×	
4.8.2 Restore Default Toolbars(AAA)	×	
4.9.2 Time-Based Media Load-Only(A)	✓	
4.9.5 Playback Rate Adjustment for Prerecorded Content(A)	×	
4.9.6 Stop/Pause/Resume Multimedia(A)	✓	
4.9.6 Navigate Multimedia(A)	✓	
4.9.7 Semantic Navigation of Time-Based Media(AA)	✓	
4.9.8 Track Enable/Disable of Time-Based Media	×	
4.9.9 Sizing Playback Viewport(AA)	×	
4.9.10 Scale and position alternative media tracks(AAA)	×	El subtitulado y la audiodescripción no van independientes de la base de tiempos.
4.9.11 Adjust Playback Contrast and Brightness	×	
5.1.1 Option to Ignore(AA)	×	
5.1.2 Retrieval Progress(A)	×	No aparece la carga del contenido (video)
5.3.1 Accessible documentation	×	
5.3.2 Document Accessibility Features(A)	×	
5.3.3 Changes Between Versions(AA)	×	
5.3.4 Centralized View(AA)	×	
5.3.5 Context Sensitive Help(AAA)	×	
5.3.6 Appropriate Language	×	
5.4.2 **Unpredictable focus**	✓	

Tabla 16: Tabla de análisis de acuerdo a las UAAG 2.0

Tal y como se observa en esta tabla, y haciendo un recuento de las directrices que cumple la página web, se alcanza un nivel de conformidad “A”.

Si repasamos la tabla, podemos observar que quedan varios criterios para alcanzar el nivel de conformidad “AA”, en concreto quedan 14 criterios para conseguirlo.