FACOLTÀ DEGLI STUDI DI TRENTO

Facoltà di Scienze Matematiche, Fisiche e Naturali

Laurea in INFORMATICA

ELABORATO FINALE

UN´IMPLEMENTAZIONE DI LODE IN HTML5

RELATORE:
prof. Marco Ronchetti

LAUREANDO:
Giorgio Pretto

ANNO ACCADEMICO 2010-2011

Indice

1 Introduzione

LODE è un sistema di videoregistrazione delle lezioni universitarie, ottimizzato per rendere la successiva visione delle stesse da parte dell´utente il più semplice possibile.

Quello che ha spinto alla creazione di un nuovo sistema per tale scopo è stata proprio la necessità di rendere tutto più semplice e meno costoso possibile. Altri sistemi in uso al tempo di creazione di LODE permettevano già la registrazione delle lezioni, ed erano per certi aspetti anche più ricchi di funzionalità. Per esempio il sistema ePresence permetteva lo streaming in diretta delle lezioni, ma ciò comportava la presenza di un server di streaming apposito e rendeva più dispensiosa la registrazione, obbligava ad avere una qualità ridotta per permettere uno streaming fluido della lezione ed inoltre complicava la fase di acquisizione.

LODE invece è nato per puntare soprattutto sulla semplicità, sia nella fase di acquisizione sia poi per la fase di visualizzazione del lavoro finito da parte dell´utente finale.

Per mantenere queste caratteristiche nel corso degli anni è chiaramente necessario mantenere il software e tutto il progetto al passo coi tempi, per questo sono stati fatti numerosi cambiamenti nel corso degli anni.

Nella fase di acquisizione si è cercato di mantenere i requisiti al minimo, sino ad arrivare a rendere necessari solamente un pc con sistema Mac OS ed una videocamera FireWire per permettere ad un semplice studente già presente in aula di eseguire la registrazione della lezione.

Una caratteristica importante di LODE è il fatto che permette di integrare all´interno della videolezione anche le eventuali slides usate dal docente durante la lezione, permettendo quindi all´utente finale di stare al passo con il docente anche quando spiega argomenti estratti dalle slides, che altrimenti nel video non sarebbero risultate chiare.

Nella fase di post-produzione è inoltre possibile apportare altre modifiche alle lezioni acquisite, ma di questo parleremo in dettaglio successivamente nel capitolo dedicato a LODE.

L´ultima fase del sistema LODE è la distribuzione delle lezioni. Questo

può avvenire in vari modi, ma la modalità più diffusa è sicuramente quella della visualizzazione online da parte dello studente. In questa modalità le lezioni vengono rese disponibili su di un server dell´università e lo studente non deve fare altro che aprire la pagina delle lezioni tramite un browser.

Questo è attualmente possibile grazie all´applicativo client realizzato in Flash. Purtroppo però Flash non risulta più essere una tecnologia portabile e accessibile facilmente a tutti, a causa del sempre più largo utilizzo di device portatili per la navigazioni, quali smartphone, tablet pc e netbook, che non sempre godono del supporto alla tecnologia Flash, a causa delle scarse risorse a disposizione in quei dispositivi.

Pertanto per mantenere LODE un sistema di semplice utilizzo da parte dell´utente finale si è cercato di sviluppare una nuova interfaccia client, che ne permettesse l´utilizzo tramite un semplice browser, senza necessità di plugin esterni.

Questo elaborato prova a proporre come soluzione al problema della portabilità con l´HTML5, il nuovissimo standard per lo sviluppo delle pagine e degli applicativi web.

Come vedremo in seguito l´HTML5 è talmente nuovo che di fatto non è ancora uno standard, ma si trova ancora in una fase di 'draft', il che significa che anche i più moderni browser non ne garantiscono il completo supporto. Ciò ha portato durante la fase di sviluppo a diversi tipi di problemi che verranno analizzati in seguito.

Nel seguito dell´elaborato analizzeremo in dettaglio sia il completo funzionamento del sistema LODE sia lo stato di sviluppo dell´HTML5, e tramite la realizzazione di una nuova interfaccia client, giudicheremo se il suo utilizzo sia una soluzione possibile al problema di portabilità e semplicità che Flash al momento non è in grado di soddisfare.

2 Il Sistema LODE

2.1 Origine di LODE

LODE (Lecture On DEmand) è un sistema per la videoregistrazione e la distribuzione delle lezioni universitarie messo a punto dal professor M. Ronchetti presso l´Università degli studi di Trento che prevede un´alta semplicità di utilizzo sia per chi registra sia per chi deve accedere alle videolezioni.

Tale sistema permette la registrazione delle lezioni tramite un semplice computer dotato di sistema operativo di tipo Mac OS X, di una telecamera, e di un microfono per una migliore qualità audio della ripresa.

All´epoca della realizzazione del sistema LODE esistevano già altri sotware simili che offrivano la possibilità di videoregistrare le lezioni, come per esempio "ePresence". Nonostante ciò si è preferito lo sviluppo di un nuovo software da zero anzichè riusarne uno preesistente per varie ragioni di tipo tecnico che ora spiegheremo.

Dopo aver provato altri tipi di sistemi per la registrazione delle lezioni, ci si è resi conto che alcune delle funzionalità che essi offrivano non erano necessarie nel nostro caso, ed eliminandole si potevano avere alcuni vantaggi interessanti. Ad esempio, trattandosi di lezioni universitarie, lo streaming in diretta non era di alcun interesse, in quanto favorirebbe l´assenza degli studenti dalle classi. Eliminando tale funzionalità si è potuto ridurre l´attrezzatura necessaria per le riprese, abbassando così il costo e il tempo necessario per effettuare una registrazione, non necessitando più di un server di streaming al momento della ripresa.

Un secondo importante fattore era poter offrire agli studenti anche la possibilità di avere le videolezioni in modalità alternative a quella on-line, rendendo quindi necessaria una struttura dei file prodotti che fosse facilmente archiviabile per la distribuzione off-line e fosse comunque poi di facile utilizzo da parte dell´utente.

LODE è stato realizzato in modo che il prodotto finale sia visualizzabile, in tutte le sue forme, tramite un semplice browser, rendendolo

quindi compatibile con tutti i maggiori sistemi operativi (Linux, Windows, Mac) e la grande maggioranza dei computer e dispositivi mobili diffusi al giorno d´oggi (smartphone,tablet e netbook).

L´eliminazione della parte di streaming delle lezioni ha permesso inoltre di introdurre alcune funzionatlità molto utili:

- La possibilità di mettere in pausa la registrazione, molto utile in caso di lezioni con esercizi all´interno, evitando quindi di registrare il tempo concesso agli studenti per risolvere gli esercizi, nel quale non avviene nulla di effettivamente utile alla lezione

- La possibilità di modificare in post-produzione la registrazione. Utile sia in caso il docente decida di eliminare parte della lezione, sia in caso si ritenga utile unire due lezioni differenti in un´unico video perchè trattanti lo stesso argomento.

- Aumento della qualità video. Non dovendo effettuare lo streaming in diretta delle lezioni non si deve più tenere conto della banda utilizzata per la trasmissione, permettendo così di alzare la qualità del video.

Le prime versioni di LODE furono sviluppate in Delphi per Windows (con l´ausilio di Java Applets e Javascript per il lato client), e in seguito portate a linux, per un più facile mantenimento, e scritte in Java.

La versione corrente è stata scritta in Java + QuickTime e sviluppata per Mac. La scelta di migrare ad un sistema Mac fu dettata dalla maggior semplicità di utilizzo di dispositivi multimediali (videocamera di acquisizione video).

Di seguito verrà descritta la struttura ed il funzionamento dell´attuale sistema in uso.

LODE permette di dividere ogni videlezioni in più parti, chiamate capitoli. Ciò permette di seguire fedelmente le classiche lezioni frontali che spesso sono supportate da delle slide. I formati PowerPoint/LibreOffice e Pdf sono infatti supportati per poter aggiungere le slides alla videolezione,

in modo da poter sincronizzare il video con la relativa slide che era presentata durante la lezione.

2.2 Struttura di LODE

Il processo di acquisizione di una lezione con LODE si divide in 3 fasi principali:

1. Acquisizione della lezione tramite l´applicativo in Java
2. Post-produzione
3. Distribuzione della lezione

Nella prima fase di acquisizione, grazie alle modifche apportate rispetto agli altri sistemi di videolezione, sono sufficienti sul posto solamente un portatile con sistema operativo Mac OS, una videocamera ed un microfono (opzionale, solo per una migliore qualità audio).

Di fatto tutto il processo di registrazione risulta molto semplice, tanto da poter essere fatto da uno studente qualsiasi presente a lezione, senza dover occupare del personale esterno appositamente.

Durante la lezione l´operatore dovrà seguire il docente nelle varie fasi di spiegazione e introdurre i vari punti di sincronizzazione con le eventuali slides usate.

La fase di post-produzione consiste principalmente nella conversione del video nel formato richiesto (finora .flv, ma per la nuova interfaccia sviluppata durante questo progetto sarà WebM) e permette di apportare delle modifiche come visto in precedenza. In questa fase si posso infatti eseguire i tagli degli esercizi o l´accorpamento di più video in un´unica lezione.

Inoltre vengono preparati degli archivi con tutto il materiale della videolezione all´interno, in modo da poter distribuire la lezione anche in modalità offline.

La distribuzione delle videolezioni può avvenire infatti tramite varie modalità:

1. Online
2. Offline differita
3. Offline

Tutte le modalità prevedono per l´utente finale di avere a disposizione un semplice browser per poter godere della lezione.

Per le modalità offline vengono preparati degli archivi contenenti delle pagine statiche in html che permettono una volta scompattati di vedere sul proprio pc in locale la videolezione desiderata. Tali archivi possono essere ditribuiti tramite CD/DVD (Offline) o scaricati direttamente dal sito di LODE (per cui è necessario ad un certo punto disporre di una connessione. Offline differita).

La modalità online è, grazie alla diffusione di connessioni sempre più veloci distribuite nel territorio, la soluzione più usata dagli utenti finali. Prevede la semplice navigazione sul sito del sistema LODE per poter visualizzare la videolezione.

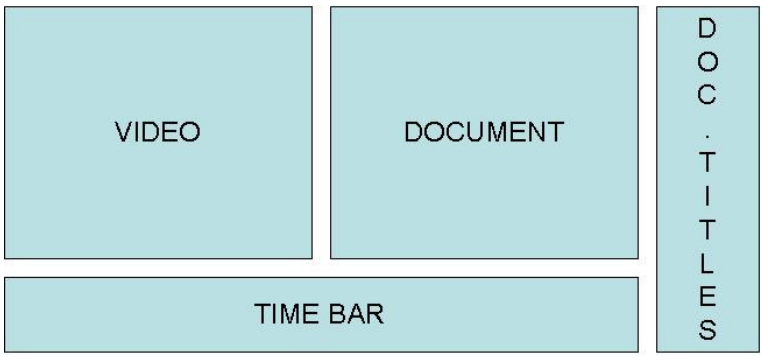
Per permettere questo è stata sviluppata un´interfaccia Web che permette all´utente di navigare all´interno della lezione grazie a degli appositi strumenti, quali una lista cliccabile di tutte le slides/capitoli in cui è divisa e una barra temporale, anch´essa divisa in varie sezioni in base alla struttura della lezione.

Lo scopo di questo elaborato è concentrarsi sull´uso da parte degli utenti della pubblicazione on-line, rendendola ancora piu' semplice e di facile utilizzo da parte di tutti, eliminando la necessità di avere un plugin Flash installato nel sistema, e agevolare quanto più possibile l´uso del sistema da parte di persone invalide.

2.3 Interfaccia On-Line del sistema LODE

Come già accennato, al giorno d´oggi grazie alla diffusione delle linee internet ad alta velocità (xDSL) il metodo di distribuzione principale è quello della visualizzazione tramite l´interfaccia online del sistema LODE, mentre la distribuzione tramite mezzi fisici (CD/DVD) è ormai in disuso, an-

che per via dell´evoluzione dei dispositivi mobili che in questi ultimi anni favoriscono l´uso di chiavette usb, eliminando in alcuni casi i lettori ottici dalle proprie dotazioni (netbook e tablet).



L´interfaccia on-line (rappresentata in figura qui sopra)prevede la visualizzazione del video e delle slides della lezione in due riquadri affiancati, ed integra inoltre degli strumenti per migliorare e facilitare la navigazione attraverso i vari 'capitoli' della lezione formati dalle slides.

Il video e la slides sono ridimensionabili per permettere allo studente di scegliere se dare più importanza alla slides o, in caso il docente stia spiegando qualcosa alla lavagna, al video.

Sulla destra troviamo una lista completa delle slides della lezione, con la quale l´utente può interagire cliccando su di esse in modo da spostarsi all´interno della lezione aggiornando in modo automatico slide e video.

Lo stesso effetto si può ottenere anche cliccando in un punto qualsiasi della time bar che si trova nella parte bassa della pagina, permettendo quindi una navigazione temporale della lezione.

Per l´implementazione in HTML5 si è scelto di mantenere la stessa interfaccia di LODE presente nell´applicativo precedente, introducendo le opportune migliorie date dalla maggior libertà dell´HTML rispetto al flash e sono inoltre state risolte alcune problematiche, come il fatto che durante il ridimensionamento delle slides e del video, queste potessero andare a coprire la time bar e l´elenco delle slides, cosa ora non più possibile.

L´eliminazione di Flash dall´interfaccia e l´utilizzo di elementi nativi ha dato molta più libertà per quanto riguarda l´interazione con l´utente e

la dinamicità della pagina. Per esempio è stato possibile introdurre le scorciatoie da tastiera, molto utili in generale, ma che specialmente rendono l´applicazione molto più accessibile da studenti con problemi di vista o disabili.

Il fatto che l´HTML5 sia un linguaggio in continua e rapida evoluzione comporta che al giorno d´oggi molte delle specifiche descritte nei siti ufficiali di WHATWG e W3C siano solamente in fase di "draft", ovvero non siano ancora arrivate allo stato di standard e pertanto in molti casi non risultano ancora supportate nemmeno dai piu´ moderni browser. Ciò ha portato al dover effettuare delle scelte importanti sul programmare seguendo le specifiche o aggirare la mancanza di supporto usando dei trucchi in Javascript, tali scelte verranno analizzate e spiegate nel capitolo relativo all´implementazione.

3 HTML5

3.1 Origine dell´HTML5

L´HTML (HyperText Markup Language) è nato nel 1990 come linguaggio per descrivere in modo semantico documenti di tipo scientifico. Da subito è stato anche usato come linguaggio di markup per le pagine del World Wide Web, e si è evolto poi fino a diventare il linguaggio odierno, adatto a descrivere molti altri tipi di documenti.

Tutte le prime revisioni del linguaggio, fino al 1995, furono ospitate prima al CERN e poi all´IETF.

Nel 1995 venne creato il W3C, che da li continuò lo sviluppo dell´HTML tramite le prime versioni 3.0 e 3.2 fino ad arrivare nel 1998 ad una versione più matura e completa: l´HTML 4.0

In quel momento i membri del W3C decisero di fermare lo sviluppo dell´HTML, e di sviluppare invece un linguaggio simile ma basato sull´XML, l´XHTML.

La prima versione altro non era che l´HTML4 riscritto in XML, senza ulteriori aggiunte eccezion fatta per la serializzazione che fu completata nel 2000.

Dopo questa prima versione l´attenzione del W3C fu quella di permettere ad altri gruppi di poter estendere il linguaggio tramite un sistema di modularizzazione, e contemporaneamente cominciò lo sviluppo dell´HTML 2.0, una nuova versione più avanzata ma non retrocompatibile.

Nel 2003 venne pubblicata una nuova tecnologia: XForms. Questo fece notare che l´XHTML era una tecnologia limitata a nuovi sviluppi (come RSS e Atom) ma non era adatta a rimpiazzare le tecnologie già esistenti.

Ciò fece rinascere l´interesse per continuare lo sviluppo dell´HTML, e Opera Software´s riuscì a dimostrare che la maggior parte delle nuove tecnologie introdotte da XForms erano implementabili in HTML4 senza dover richiedere cambiamenti da parte dei browser.

L´idea che lo sviluppo dell´HTML dovesse riprendere fu portata al W3C workshop nel 2004 da Mozilla e Opera assieme, ma i membri del W3C ritennero fosse più giusto continuare con lo sviluppo di linguaggi basati su XML, come stabilito in precedenza.

Nonostante questa decisione, poco dopo Mozilla Opera ed anche Apple si unirono creando il Web Hypertext Application Technology Working Group, o WHATWG, creando una mailing list pubblica e un sito dove pubblicare le varie bozze del linguaggio.

I principi base su cui si poggiò il WHATWG per lo sviluppo di quello che diventerà l´HTML5 furono:

- Le tecnologie dovevano essere retrocompatibili
- Le specifiche e le implementazioni dovevano combaciare
- Le specifiche dovevano essere molto dettagliate

L´ultimo requisito in particolare richiedeva che le specifiche fossero chiare in modo da permettere l´iteroperabilità tra le varie implementazioni, e ciò richiedeva un livello di dettaglio mai visto prima.

Nel 2006, nonostante il rifiuto iniziale, il W3C espresse il desiderio di partecipare allo sviluppo dell´HTML5 e nel 2007 formò un working group da affiancare al WHATWG per lo sviluppo delle specifiche. Apple Mozilla e Opera permisero al W3C di rilasciare le specifiche sotto il suo copriright, mantenendo comunque una loro versione con licenza meno restrittiva sul loro sito.

Da allora i due gruppi collaborano assieme allo sviluppo.

Nonostante ciò ad oggi esistono due versioni dell´HTML5 a causa del diverso ciclo di rilascio tra W3C e WHATWG. Il W3C prevede delle versioni di rilascio con il classico ciclo di “feature freeze”, ovvero una volta stabilite le funzionalità da rilasciare con una certa versione (in questo caso la 5) aspetta che vengano stabilizzate e nel frattempo blocca la possibilità di ulteriori aggiunte alle specifiche.

Al contrario il WHATWG non prevede versioni, di fatto per loro si parla solo di HTML, senza alcun numero. Ciò permette alle nuove funzion-

alità di essere pubblicate più velocemente nelle specifiche, anche se non completamente stabili e passibili di future modifiche.

3.2 Novità dell´HTML5

L´HTML5 introduce molte nuove funzionalità rispetto alla precedente versione, e introduce nuovi tag anche per semplificare lo sviluppo ai programmatori web.

Molti di questi nuovi tag sono dettati principalmente dalle esigenze del web, che sta diventando sempre più ricco di contenuti multimediali, e di siti personali, al giorno d´oggi composti per la maggior parte da blog e siti di notizie.

Per esempio sono stati introdotti i tag <header> e <footer>, che rappresentano due parti presenti in ogni sito, e che fino ad ora dovevano essere create con dei e delle classi specificate poi nei css. Questi nuovi tag possono essere usati più volte nella stessa pagina, permettendo per esempio in un sito con degli articoli, di creare una sezione d´intestazione contenente i dati relativi all´articolo, come autore, data ed altro. Ciò permette di avere un codice meglio organizzato.

Altri tag al riguardo sono proprio <author> <time> e <article>, che possono essere usati proprio per specificare un articolo, il suo autore e la data di pubblicazione all´interno di un sito.

I tag descritti fin´ora di fatto non aggiungono funzionalità all´HTML5, ma permettono di creare una migliore struttura delle pagine web, migliorando le caratteristiche di tipo SEO (Search Engine Optimization), rendendo la pagina più facile da cercare nel web. Inoltre i vantaggi di avere un specifico tag per queste funzioni anzichè usare dei semplici o è quello che un tag può avere dei propri parametri. Questo permette di inserire ulteriori informazioni tramite gli attributi, per esempio in un tag <time> possiamo specificare la TimeZone, o sempre per restare in tema, se viene specificato l´attributo 'pubdate' si specifica che la data è una data di pubblicazione di un articolo.

Fin qui abbiamo visto solo migliorie ma nessunà novità effettiva.

Come specificato nel primo capitolo, l´HTML5 è nato grazie alla pubblicazione della tecnologia XForms, ed è pertanto logico aspettarsi che gli sviluppi maggiori siano stati fatti riguardo alle form html, come di fatto è.

Sono state infatti introdotte molte migliorie e novità per quanto riguarda le form, specialmente per per il tag <input>, che è ora molto più completo e versatile.

Sono stati introdotti nuove possibilità per l´attributo type, quali telephone, e-mail, month, week e molti altri. Questo permette di avere un maggior controllo sui dati inseriti dall´utente, specificando in anticipo che tipo di dato andrà inserito in ogni campo, anzichè del generico text come in precedenza.

Questo facilita di molto anche la procedura di validazione dell´input. Assieme ad altri attributi come required, che permette di non inviare il form se il campo non è specificato, ed altri come min e max che a loro volta non permettono di essere sicuri che l´utente non inserisca valori errati, queste nuove tipologie di input permettono di eseguire dei controlli di validazione in maniere molto più semplice senza dover ricorrere a Javascript, rendendo quindi il tutto anche molto più veloce.

Sicuramente le novità più importanti sono quelle che riguardano la multimedialità di una pagina web.

Parliamo ora dei tag <video> e <audio> che permettono di introdurre in una pagina dei contenuti multimediali in modo nativo e semplice, senza dover ricorrere all´utilizzo di plugin esterni ad un browser, quali Flash o SilverLight, che era fino ad ora l´unica soluzione.

Il tag <audio> permette di aggiungere in una pagina html una traccia sonora. È possibile per l´autore della pagina rendere visibili dei controlli per l´utente o meno. In caso di tracce audio da ascoltare per esempio è logico mostrare i controlli, in modo che il visitatore della nostra pagina possa avere controllo sulla pagina. Se invece le tracce audio le vogliamo utilizzare con altri scopi, come per esempio degli effetti sonori, o sottofondo musicale, non è detto che vogliamo lasciare il controllo all´utente, anzi.

Per il tag <video>, dato che si tratta dell´argomento principale del nostro progetto, svilupperemo un capitolo apposito.

3.2.1 I video in HTML5

Il tag <video> viene utilizzato per includere nelle pagine web elementi come film, composti da una traccia video e un´opzionale traccia audio. Assieme al tag <audio> completano la famiglia dei Media Elements dell´HTML5, pertanto è realizzato tramite l´implementazione dell´interfaccia HTMLMediaElement .

Oltre ad implementare l´interfaccia HTMLMediaElement, il tag <video> la estende con una sua interfaccia che prevede l´aggiunta di ulteriori parametri, per ulteriori dettagli vedere il prossimo capitolo.

All´interno del tag video può essere specificato l´attributo src, ovvero un url che indica la sorgente video da mostrare nella pagina. È importante notare come tale tag non sia obbligatorio come lo è invece per le immagini. Questo, nonostante possa sembrare strano, è invece indispensabile per la compatibilità della nostra pagina con tutti i browser, grazie ad un altro importante tag, il tag <source>.

Putroppo come spesso accade in questo genere di cose, l´introduzione del tag <video> ha portato da subito ad una guerra dei codec, per stabilire quale formato video fosse il migliore per la pubblicazione di video, generando di fatto confusione e ad oggi ancora non c´è un formato unico per tutti i browser, anche se WebM sembra lo diventerà a breve.

Proprio per risolvere questo tipo di problema è stato introdotto il tag <source> che permette di specificare per un singolo video più sorgenti, semplicemente inserendole una dopo l´altra dentro il tag <video>. Il browser tenta di visualizzare la prima sorgente, passando alla successiva in caso non riesca nella riproduzione della traccia specificata. Questa tecnica permette non solo di supportare più formati video, ma tramite la funzione 'onerror' sull´ultima traccia possiamo anche lanciare una nostra funzione in caso il browser non sia capace di visualizzare nessuna delle tracce. Questo metodo è usato anche nel nostro progetto per implementare il sis-

tema di fallback alla vecchia versione in Flash in caso il browser dell´utente ancora non supporti a nuova interfaccia LODE in HTML5.

Un altro tag importante per sfruttare appieno le funzionalità dei video in HTML5 è il tag <track> che permette di integrare il video con delle tracce per i sottotitoli e non solo.

Tale tag permette di specificare delle 'Text Tracks' che verranno sincronizzate con il video. Tali tracce possono essere di tipo 'subtitles', 'captions', 'descriptions', 'chapters', 'metadata'.

Le prime quattro tipologie servono ad aggiungere del testo al video, permettendo oltre alla visualizzazione dei sottotitoli anche l´aggiunta di descrizioni per alcune scene, o di mantenere presente una stringa che descriva il capitolo che stiamo visualizzando, o ancora di aggiungere delle descrizioni particolari per non udenti, rendendo i contenuti del nostro sito più accessibile.

Particolarmente interessante risulta essere il tipo di traccia 'metadata'. Questo tipo di traccia è stata introdotta non per essere visualizzata dal browser, ma per essere eseguita come uno script. Le possibilità che questo tipo di traccia da al programmatore sono molte, in particolar modo in un progetto come il nostro, ma vedremo poi nel capitolo 3.3 come questo putroppo non sia avvenuto.

3.3 Dettagli delle specifiche

Di seguito verranno riportate e descritte le specifiche relative agli elementi più importanti usati durante lo sviluppo della nuova interfaccia HTML5 di LODE

3.3.1 Descrizione dell´interfaccia HTMLVideoElement

```
interface HTMLVideoElement : HTMLMediaElement {
attribute unsigned long width;
attribute unsigned long height;
readonly attribute unsigned long videoWidth;
readonly attribute unsigned long videoHeight;
attribute DOMString poster;
};
```

width Indica la larghezza dell´elemento all´interno della pagina

height Indica l´altezza dell´elemento all´interno della pagina

videoWidth Indica la larghezza intrinseca del video, ovvero la risoluzione nativa orizzontale.

videoHeight Indica l´altezza intrinseca del video, ovvero la risoluzione nativa verticale

poster Stringa contenente il link ad un´immagine da usare quando il video deve ancora partire. Solitamente un Frame del video

Va sottolineato che mentre videoWidth e videoHeight rappresentano sostanzialmente la risoluzione nativa del video, e non possono perciò essere modificati, width ed height indicano le dimensioni fisiche dell´oggetto nella pagina, e possono essere modificate a piacimento, anche se ciò può portare a deformazioni del video.

3.3.2 Descrizione dell´Interfaccia HTMLMediaElement

```
interface HTMLMediaElement : HTMLElement {
// error state
readonly attribute MediaError error;
// network state
attribute DOMString src;
```

```
readonly attribute DOMString currentSrc;
const unsigned short NETWORK_EMPTY = 0;
const unsigned short NETWORK_IDLE = 1;
const unsigned short NETWORK_LOADING = 2;
const unsigned short NETWORK_NO_SOURCE = 3;
readonly attribute unsigned short networkState;
attribute DOMString preload;
readonly attribute TimeRanges buffered;
void load();
DOMString canPlayType(in DOMString type);
// ready state
const unsigned short HAVE_NOTHING = 0;
const unsigned short HAVE_METADATA = 1;
const unsigned short HAVE_CURRENT_DATA = 2;
const unsigned short HAVE_FUTURE_DATA = 3;
const unsigned short HAVE_ENOUGH_DATA = 4;
readonly attribute unsigned short readyState;
readonly attribute boolean seeking;
// playback state
attribute double currentTime;
readonly attribute double initialTime;
readonly attribute double duration;
readonly attribute Date startOffsetTime;
readonly attribute boolean paused;
attribute double defaultPlaybackRate;
attribute double playbackRate;
readonly attribute TimeRanges played;
readonly attribute TimeRanges seekable;
readonly attribute boolean ended;
attribute boolean autoplay;
attribute boolean loop;
void play();
void pause();
// media controller
attribute DOMString mediaGroup;
attribute MediaController controller;
// controls
attribute boolean controls;
```

```
    attribute double volume;
    attribute boolean muted;
    attribute boolean defaultMuted;
    // tracks
    readonly attribute MultipleTrackList audioTracks;
    readonly attribute ExclusiveTrackList videoTracks;
    readonly attribute TextTrack[] textTracks;
    MutableTextTrack addTextTrack(in DOMString kind, in optional DOMString
label, in optional DOMString language);
    };
```

error è un attributo di tipo MediaError e indica, in caso di errore del video, che tipo di errore si è verificato.

src indica la sorgente del video. Può essere settata quando si definisce il tag <video> o in seguito tramite javascript. Deve essere una stringa con il percorso ed il nome del video.

currentSrc come Src, indica la sorgente del video, ma CurrentSrc non può essere settata direttamente dall´utente poichè viene calcolata in modo dinamico dal browser, che controlla se il video viene caricato dall´attributo Src o, in caso quest´ultimo sia nullo, dal tag <source> che può essere annidato nel tag video in caso si voglia specificare più file, come nel caso di codifiche differenti (.webm, .ogg, . . .).

networkState indica lo stato della connessione del video. Può assumere solo i valori prefissati dalle costanti definite nell´interfaccia

- ⇀ NETWORK_EMPTY = 0
- ⇀ NETWORK_IDLE = 1
- ⇀ NETWORK_LOADING = 2
- ⇀ NETWORK_NO_SOURCE = 3

readyState indica lo stato del video, basandosi su quanti dati ha caricato sino a quel momento. Può assumere solo i valori prefissati dalle costanti definite nell´interfaccia

- HAVE_NOTHING = 0
- HAVE_METADATA = 1
- HAVE_CURRENT_DATA = 2
- HAVE_FUTURE_DATA = 3
- HAVE_ENOUGH_DATA = 4

preload serve ad indicare al browser se iniziare il download del video al caricamento della pagina, e può assumere solo i seguenti valori : none, metadata, auto. Con none, il browser non deve caricare nulla del video, con metadata devono essere caricati solo i metadati (come la lunghezza del video, le dimensioni, . . .), mentre se viene passato il parametro auto allora il video deve cominciare a scaricare il video immediatamente.

buffered è un attributo di tipo TimeRanges e serve ad indicare l´attuale range del video che è bufferizzato dal browser.

seeking è un attributo di tipo booleano che indica se in quel momento il video è in una fase di seek, ovvero se l´utente ha richiesto la visualizzazione di una parte diversa da quella attualmente in esecuzione.

currentTime indica, in secondi, la posizione di riproduzione attuale del video.

initialTime indica la posizione di partenza iniziale per la riproduzione del video, solitamente 0.

duration indica, in secondi, la durata totale del video.

startOffsetTime è un attributo che ritorna un oggetto di tipo Date rappresentante il timeline offset attuale, ovvero la data che corrisponde al tempo di inizio del video (solo alcuni vido supportano questo attributo).

paused è un attributo di tipo booleano che indica se il video è in pausa.

defaultPlaybackRate tipicamente settato al valore 1.0, indica la velocità standard di riproduzione del video, intesa come un moltiplicatore (Es: 2.0 va il doppio più veloce).

playbackRate indica la velocità di riproduzione del video, settata inizialmente al valore di defaultPlaybackRate può essere alterata tramite javascript.

played è un attributo di tipo TimeRanges e contiene la parte di timeline del video che è stata riprodotta.

seekable è un TimeRanges che sta ad indicare quale parte del video è accessibile all´utente tramite un´azione di seek.

ended è un valore booleano che viene settato a true se la riproduzione del video raggiunge la fine del video stesso.

autoplay è un attributo booleano che può essere settato nella dichiarazione del video, e indica al browser se al caricamento del video la riproduzione deve essere avviata automaticamente.

loop è un attributo booleano che indica, se settato a true, che una volta raggiunta la fine, il video deve ricominciare la riproduzione dall´inizio.

mediaGroup è un attributo usato per collegare assieme tra loro più media elements creando automaticamente un mediaController

mediaController è un attributo di tipo MediaController. Un MediaController viene usato per tenere sincronizzati tra loro vari media elements.

controls è un attributo booleano che indica al browser se deve mostrare o meno i controlli nativi per la gestione del video.

volume è un moltiplicatore che accetta valori tra 0.0 e 1.0 che serve a modificare il livello del volume del video.

muted è un valore booleano che segnala quando tutte le sorgenti audio del media element sono mute.

defaultMuted è un valore booleano che riflette lo stato dell´attributo muted.

audioTracks è un attributo che contiene un oggetto di tipo MultipleTrackList rappresentante le traccie audio disponibili nel media element.

videoTracks è un attributo che contiene un oggetto di tipo ExclusiveTrackList che rappresenta le traccie video disponibili all´interno del media element.

textTracks ritorna un array di oggetti di tipo TextTrack. È di lunghezza fissa e di sola lettura, dato che la lista di tracce viene generata tramite il tag <track> annidato nel tag <video>.

Descrizione dei metodi dell´interfaccia.

load() si occupa di ricaricare la sorgente video, che sia cambiata o meno, scartando tutte le operazioni in sospeso sulla vecchia traccia.

canPlayType(type) restituisce una stringa vuota in caso il browser sappia di non poter riprodurre quel tipo ti traccia, altrimenti restituisce la stringa “probably” se pensa di riuscire a riprodurre la traccia nei tag <audio> o <video>, infine deve ritornare “maybe” altrimenti.

play() riprende la riproduzione del video.

pause() sospende la riproduzione del video.

addTextTrack(kind,[label,[language]]) crea una nuova textTrack con i parametri passati e poi la aggiunge alle tracce già presenti nel video.

4 Implementazione in HTML5

4.1 Vantaggi di un´interfaccia in HTML5

L´attuale interfaccia web del sistema LODE è realizzata tramite un´applicazione in Flash, il che richiede all´utente di avere a disposizione oltre ad un browser anche un ulteriore plugin che aggiunga il supporto a Flash.

Quando è stato scelta Flash come tecnologia di sviluppo Flash era una delle poche realtà che garantiva il supporto a tutte le piattaforme, rendendo il lato client di LODE di semplice utilizzo da parte dell´utente finale e molto portabile.

Recentemente però Flash risulta non essere più una tecnologia multipiattaforma e compatibile come lo è stato negli anni passati. Flash risulta problematico per le piattaforme a 64 bit, in quanto ancora oggi per il sistema operativo Linux e simili non esiste un player Flash nativo a 64 bit, ma si devono usare vari trucchi e plugin per far funzionare le versioni a 32 bit, il che spesso porta a dei problemi anche gravi (crash del browser o plugin Flash che usa il 100% delle risorse della cpu).

Moltissime persone inoltre utilizzano sempre più spesso dei dispositivi per la navigazione che non garantiscono il supporto a Flash, basti pensare agli smartphone, o ai nuovi tabled come l´iPad, che per scelta in alcuni casi decidono di non supportare la tecnologia di Adobe che finora ha dominato l´aspetto multimediale del web.

Con l´introduzione dei nuovi tag dell´HTML5 è possibile riscrivere l´applicazione eliminando del tutto Flash, rendendo quindi non necessaria la presenza di un ulteriore plugin per l´utente finale, rendendo l´interfaccia fruibile da un semplice browser con supporto ai JavaScript.

Questo permette di avere una pagina web vera e propria, ricca di elementi multimediali nativi, con cui è possibile comunicare e operare tramite Javascript, permettendo un controllo totale da parte del programmatore sulle varie parti dell´applicazione, e rendendo molto più facili interventi di mantenimento o modifica dell´interfaccia. Con la precedente versione invece l´applicazione era molto meno modulare, e dal punto di vista del

codice HTML era solo un grosso oggetto importato nella pagina.

4.2 Scelte Implementative

Durante lo sviluppo dell´applicazione ci si è trovati di fronte più volte al problema di cosa fare quando le specifiche HTML5 risultavano non supportate dai maggiori browser.

La linea che si è cercato di seguire è stata quella di seguire le specifiche HTML5 dove ciò fosse possibile, senza cercare di ovviare ai problemi risolvendoli con trucchi in Javascript, che avrebbero non solo snaturato l´obiettivo del progetto di creare un´applicazione più semplice e aderente agli standard possibile, ma anche appesantito l´interfaccia.

Un problema in cui ci si è imbattuti subito è stato la completa mancanza di supporto per il tag <track> da parte di tutti i browser, e perfino dello stesso validatore della W3C in quanto tale tag risulta estremamente recente.

Come descritto ancora nel primo capitolo tale tag avrebbe permesso di utilizzare un sistema simile a quello per i sottotitoli, ma per la sincronizzazione di script rispetto al video. Sarebbe stato possibile cambiare le slides in modo automatico durante la riproduzione del video, senza doverne controllare periodicamente lo stato di avanzamento da una funzione esterna. La soluzione in uso attualmente difatti si basa sull´evento 'onprogress' del tag <video> che lancia un evento ad ogni visualizzazione di un frame, il che per una nostra videolezione standard sta a significare circa 12 volte al secondo. Ciò ha reso necessario implementare tale funzione con i controlli minimi necessari, altrimenti sarebbe risultato troppo pesante per un un pc non troppo recente.

Inoltre il tag <track> era essenziale per migliorare ulteriormente l´interfaccia web di LODE riguardo il tema dell´accessibilità. Avrebbe infatti permesso l´uso di sottotitoli per le lezioni, o di aggiungere il titolo o il testo delle slide direttamente nel video.

Come detto in precedenza, in questo caso, si sarebbe dovuto procedere comunque all´implementazione di un sistema che usasse i tag <track>, ma

chiaramente, essendo che nessun browser testato, e perfino la mailing list del W3C abbia confermato il totale non supporto del tag, ha reso impossibile ogni tipo di testing al rigardo.

Fortunatamente per molte delle altre funzionalità richieste dal nostro applicativo si riusciva a trovare almeno un browser che supportasse tale funzione (nella maggior parte dei casi Chrome/Chromium) permettendo quindi di procedere con l´implementazione di tale funzionalità e di fare i relativi test per verificarne il funzionamento.

Simili situazioni si sono verificate nella programmazione delle funzioni di modifica della velocità di riproduzione del video (playbackRate). Secondo le specifiche modificando tale valore del video, la velocità del video dovrebbe modificarsi di conseguenza, comportamento riscontrato veritiero solamente su Chromium. Ma il fatto che almeno su di un browser fosse possibile eseguire dei testing ha permesso di completarne lo sviluppo, in attesa che anche gli altri browser si aggiornino e si adeguino agli standard. Da sottolineare che in realtà nemmeno Chromium supporta a pieno la funzione, in quanto dopo alcuni test ci si è resi conto che la modifica della velocità di riproduzione funziona solo per valori positivi, al contrario di quanto dettato dalle specifiche, ma questo in realtà è ininfluente, in quanto generalmente non ha alcun senso guardare una videolezione all´indietro.

Un altro caso simile si è presentato nell´implementazione del buffer video, ovvero quando andiamo a mostrare all´utente nella time bar dell´interfaccia quanto è stato caricato il video. Anche in questo caso la funzionalità risulta avere un comportamento corretto solamente su Chromium, e anche qui non totalmente.

Si spera che i browser si adegueranno con il tempo alle specifiche dettate dallo standard HTML5 rendendo le funzioni effettivamente usabili ed uguali per tutti. Questo putroppo non è così scontato, dato che spesso le case produttrici dei browser quando implementano funzioni che ancora non sono diffuse e standard inventano tag con nomi diversi. Un chiaro esempio di questo è il tag <itext> supportato da Mozilla che implementa le stesse funzionalità (circa) del tag <track>.

4.3 Possibili miglioramenti

Nella nuova interfaccia LODE in HTML5 è possibile fare ancora delle migliorie, per la maggior parte legate come detto nel capitolo precedente alle specifiche non ancora supportate.

Altri miglioramenti riguardano il miglior uso di elementi nativi dell´HTML5.

Al momento per lo sviluppo dell´applicativo ci si è appoggiati alla libreria MooTools per tutta la parte relativa a JavaScript, e ciò ha portato ad adattarsi alle funzioni implementate da tale libreria anche se in certe circostanze sarebbe stato meglio usare strumenti più adatti.

Nel caso degli slider per esempio, usati nell´interfaccia di LODE sia per il ridimensionamento del video e della slide, sia per la realizzazione della barra temporale, risulterebbe forse più opportuno usare un elemento input di tipo range, di recente introduzione. Tale elemento però può essere visualizzato dal browser in vari modi, di fatto nelle specifiche stesse non è scritto nulla al riguardo della visualizzazione grafica dell´elemento, pertanto il fatto che ogni browser possa inventarsi una rappresentazione diversa per lo stesso elemento potrebbe portare a dei problemi nella visualizzazione finale dell´interfaccia.

Inoltre non è stato testato se tale elemento sia modificabile tramite JavaScript con la stessa facilità e libertà richieste per l´attuale implementazione degli sliders. Per questi motivi si è deciso di mantenere le librerie esistenti.

Un aspetto molto importante da considerare per un futuro miglioramento dell´applicazione è l´accessibilità.

Rispetto alla precedente versione sono già stati fatti molti passi avanti, essendo state introdotte le scorciatoie da tastiera che permettono di realizzare le seguenti funzioni senza l´utilizzo del mouse:

- Play / Pause
- Ridimensionamento Video / Slide
- Navigazione alla slides successiva o precedente
- Navigazione temporale in avanti o indietro nel video

I comandi dell´interfaccia sono stati realizzati di dimensioni maggiori rispetto a prima, per permettere una maggiore visibilità dei controlli anche ad ipovedenti. Inoltre l´utente può scegliere un tema di colori ad alto contrasto, sempre per permetter una maggiore visibilità della pagina.

Nonostante questo appena il tag <track> sarà supportato si potranno aggiungere sottotitoli per i video e anche trascrizioni complete di slides, anche se per fare questo non sarà possibile toccare solo la parte dell´interfaccia web di LODE ma si dovranno fare delle aggiunte anche nella fase di post-produzione per creare le informazioni aggiuntive da passare poi all´applicativo web.

5 Conclusioni

L´HTML5 si è rivelato essere sulla carta un´ottima soluzione per quanto riguarda sia la portabilità del client sia per la facilità d´uso da perte dell´utente finale.

Le nuove funzionalità descritte nelle specifiche permettono infatti di replicare tutte le funzioni già esistenti nell´attuale client di LODE, rendendole addirittura di più facile utilizzo e perfino maggiormente accessibili.

Questo però non vale per tutte le funzionalità dell´applicativo.

Difatti, come è stato riscontrato nella fase di sviluppo del progetto, al giorno d´oggi non tutti i maggiori browser supportano a pieno le funzionalità introdotte da HTML5, rendendo di fatto l´applicativo non funzionante al 100%, sebbene l´utilizzo base del sistema LODE sia comunque garantito.

Il nuovo client sviluppato per questo elaborato risulta quindi essere già un buon sostituto per l´attuale applicativo sviluppato con tecnlologia Flash, mantenendo ancor un buon margine di miglioramento per quanto riguarda funzionalità ed accessibilità, difatti con i browser maggiormente diffusi al momento (Opera, Firefox, Chrome, IE) l´utente finale può tranquillamente godere della videolezione, sebbene in alcuni casi in maniera leggermente limitata.

HTML5 è uno 'standard' in costante sviluppo e aggiornamento, ma così sono anche i browser.

In un futuro prossimo ci si aspetta quindi che le funzionalità richieste dal client LODE in HTML5 per un´esperienza più ricca da parte dall´utente vengano implementate correttamente e permettano lo sviluppo dei sottotitoli e delle altre funzionalità aggiuntive che non è stato possibile aggiungere ora, rendendo l´applicativo sempre più completo.

Il nuovo client come già spiegato nel capitolo relativo all´implementazione è già predisposto con alcune funzioni che si attiveranno automaticamente appena i browser lo permetteranno. Quello che resta da fare in futuro sarà rendere ancora più accessibile l´applicazione sfruttando tutte le risorse che l´HTML5 metterà a disposizione.

Concludendo, l´interfaccia in HTML5 del client LODE risulta essere una buona soluzione e promette di restare tale anche in futuro, mantenendo le caratteristiche di semplicità ed accessibilità che erano richieste.

Appendice

In appendice vengono allegati i listati del codice sorgente delle pagine principali del progetto.

Index.html

```
<!DOCTYPE html>
<html lang='en'>
<head>
<title>Lecture On DEmand HTML5- University of Trento</title>
<link id='styleCss' rel='stylesheet' href='./css/style.css'
type='text/css' media='screen'>
<link id='colorCss' rel='stylesheet' href='./css/color/macOs.css'
type='text/css' media='screen'>
<meta charset='UTF-8'>
<meta name='application-name' content='LODE'>
<meta name='author' content='Giorgio Pretto'>
<meta name='author' content='Università di Trento'>
<meta name='description'
content="LODE è un sistema di Lecture On DEmand sviluppato
dall'Università di Trento">
<meta name='keywords'
content='LODE,lode,lecture on demand,lezioni on line,università,
trento,università di trento,unitn,videolezioni'>
<script src='js/mootoolsCore-1.3.js' type='text/javascript'></script>
<script src='js/mootoolsMore-1.3.js' type='text/javascript'></script>
<script src='js/mooLODE/LODE.js' type='text/javascript'></script>
<script src='js/init.js' type='text/javascript'></script>
</head>
<body>
<noscript>
<div id="noscript">
You have Javascript disabled!  Enable it to activate all the site functions
<br/>
You can still watch the video with Javascript disabled
</div>
</noscript>
<header>
<div>
<img id="logo" src="loghi/logoLODE.svg" alt="LODE - Lecture On DEmande">
<section id="info">
Corso :  <span id="corso"></span><br/>
Titolo :  <span id="titolo"></span><br/>
Docente :  <span id="docente"></span><br/>
</section>
</div>
</header>
<section id="interface">
<div>
<ol id='slides'>
<li>loading slides...
</ol>
<div id="resizeControls">
<button id="bigSlide">BIG SLIDE</button>
<button id="bigVideo">BIG VIDEO</button>
<div id='resizeContainer'>
<div id='resizeHandle'
```

```
title="Drag to resize Video and Slide">
RESIZE
</div>
</div>
</div>
<section id='main'>
<a id="slideLink" href="img/1.jpg" target="_blank">
<img id='slide' src="img/1.jpg" alt="">
</a>
<video id='video' preload controls>
<source src='movie.webm' onerror="fallback()">
</video>
</section>
<section id='controls'>
<div id="rightControls">
Volume :
<div id='volumeContainer'>
<div id='volumeHandle'>1.0</div>
</div>
Speed :
<div id='playrateContainer'>
<div id='playrateHandle'>1.0</div>
</div>
</div>
<div id="leftControls">
<span id="currentTimeLabel">0:00:00</span>
-
<span id="totalTimeLabel">0:00:00</span>
</div>
<button id="rew">REW</button>
<button id="playPause">PLAY<br/>PAUSE</button>
<button id="ff">FF</button>
<div id="buffered"></div>
<div id='seekContainer'>
<div id="seekSlides"></div>
<div id='seekHandle'></div>
</div>
</section>
</div>
</section>
<footer>
<div>
<section id="tips">
<ul>
<li>Space = Play / Pause
<li>Control+right = Fast Forward
<li>Control+left = Rewind
<li>Control+up = Previous Slide
<li>Control+down = Next Slide
<li>Shift+right = Enlarge Video
<li>Shift+left = Enlarge Slide
</ul>
</section>
</div>
</footer>
<div id="redirect">
Your browser doesn't seems to support the new HTML5/WebM LODE interface<br/>
You'll be now redirect to the old Flash version in 3s
<br/><br/>
Sorry 'bout that :(
</div>
</body>

</html>
```

style.css

```
body {
font-family:  "Verdana", "Helvetica", monospace;
}
/*### BUTTONS STYLE ####*/
button {
box-shadow:  0 0 3px 2px #CCF;
cursor:  pointer;
}
button:hover{
box-shadow:  0 0 4px 1px #369;
}
button:active{
box-shadow:  inset 0 0 5px #369, 0 0 4px 3px #369;
}
/*#### HEADER and FOOTER ####*/
header {
width:  100%;
padding:  10px 0 0 0;
margin:  0;
overflow:  hidden;
}
header > div {
width:  1000px;
padding:  5px 0 5px 10px;
margin:  0 auto;
box-shadow:  0 0 30px #FFF;
}
footer {
width:  100%;
padding:  0 0 20px 0;
margin:  0;
overflow:  hidden;
display:  inline-block;
}
footer > div {
width:  1000px;
padding:  5px 0 5px 10px;
margin:  0 auto;
box-shadow:  0 0 30px #FFF;
}
#logo {
height:  50px;
float:  left;
}
#info {
display:  inline-block;
margin:  0 0 0 55px;
font-weight:  bold;
font-size:  .8em;
}
#info span {
font-weight:  normal;
}
#interface {
overflow:  hidden;
}
#interface > div{
width:  1000px;
padding:  0 0 20px 10px;
margin:  0 auto;
box-shadow:  0 0 30px #666;
```

```
}
/*### MAIN Video + Slides ####*/
#main {
display:  inline-block;
width:  800px;
clear:  both;
box-shadow:  0px 0px 5px 1px rgba(0,0,0,0.4);
}
#video {
width:  399px;
float:  left;
}
#slide {
width:  399px;
float:  right;
cursor:  pointer;
}
/*#### SLIDES - side column####*/
#slides {
margin:  0;
float:  right;
width:  180px;
padding:  10px 5px 20px 5px;
box-shadow:  -1px 1px 5px #369;
}
#slides > li{
font-size:  0.7em;
margin:  2px;
list-style:  decimal-leading-zero;
list-style-position:  inside;
}
#slides > li:hover{
cursor:  pointer;
}
/*##### CONTROLS ####*/
/*# RESIZE CONTROLS #*/
#resizeControls {
display:  inline-block;
width:  800px;
height:  24px;
padding:  4px 0;
margin:  5px 0;
}
#resizeContainer {
width:  620px;
height:  16px;
border-radius:  16px;
box-shadow:  inset -3px 2px 3px rgba(0,0,0,0.4);
margin:  4px 90px 0 90px;
overflow:  hidden;
}
#resizeHandle {
width:  60px;
height:  16px;
cursor:  move;
border-radius:  16px;
box-shadow:  -1px 0 3px 1px #369;
text-align:  center;
font-weight:  bold;
font-size:  0.7em;
padding:  0;
position:  relative;
left:  280px;
}
#resizeHandle:hover{
```

```
box-shadow:  0 0 4px 1px #369;
}
#bigVideo, #bigSlide {
width:  80px;
height:  16px;
border:  0px solid;
font-size:  0.7em;
font-weight:  bold;
font-variant:  small-caps;
margin:  4px 0;
padding:  0;
border-radius:  16px;
}
#bigVideo {
float:  left;
}
#bigSlide {
float:  right;
}
/*# VIDEO CONTROLS #*/
#controls {
width:  800px;
margin:  5px 0 0 0;
display:  inline-block;
padding:  0;
text-align:  center;
}
/* Seek Slider */
#seekContainer {
width:  800px;
height:  15px;
border-radius:  15px;
box-shadow:  inset -3px 2px 3px rgba(0,0,0,0.4);
margin:  0;
overflow:  hidden;
}
#seekHandle {
width:  15px;
height:  15px;
margin:  -15px 0 0 0;
box-shadow:  -3px 0px 2px rgba(0,0,0,0.4);
border-radius:  15px;
cursor:  move;
}
#rightControls {
float:  right;
width:  210px;
text-align:  right;
}
#leftControls {
float:  left;
width:  210px;
text-align:  left;
font-weight:  bold;
font-size:  1.2em;
padding:  5px 0;
}
#playrateContainer {
display:  inline-block;
width:  80px;
height:  15px;
border-radius:  15px;
box-shadow:  inset -3px 2px 3px rgba(0,0,0,0.4);
margin-top:  10px;
overflow:  hidden;
```

```
}
#playrateHandle {
text-align:  center;
font-family:  "Verdana",sans-serif;
font-size:  0.7em;
font-weight:  bold;
border-radius:  15px;
box-shadow:  -3px 0px 2px rgba(0,0,0,0.4);
width:  30px;
height:  15px;
cursor:  move;
}
#buffered {
margin:  5px 0 -15px 0;
height:  15px;
width:  40px;
border-radius:  15px;
opacity:  .5;
}
#seekSlides {
padding:  0;
margin:  0 0 0 7px;
border-right:  2px solid;
width:  785px;
height:  15px;
text-align:  left;
}
.seekSlide {
float:  left;
margin:  0;
border-left:  2px solid;
height:  25px;
opacity:  0.5;
}
.seekSlide:hover {
cursor:  pointer;
}
/*volume settings*/
#volumeContainer {
display:  inline-block;
width:  80px;
height:  15px;
border-radius:  15px;
box-shadow:  inset -3px 2px 3px rgba(0,0,0,0.4);
overflow:  hidden;
}
#volumeHandle {
text-align:  center;
font-family:  "Verdana", sans-serif;
font-size:  0.7em;
font-weight:  bold;
border-radius:  15px;
box-shadow:  -3px 0px 2px rgba(0,0,0,0.4);
width:  30px;
height:  15px;
cursor:  move;
}
/* ## buttons ##*/
#controls > button {
font-size:  0.7em;
font-weight:  bold;
}
#playPause {
width:  50px;
height:  50px;
```

```
border: 0px;
border-radius: 50px;
}
#rew, #ff {
width: 40px;
height: 20px;
border: 0px;
border-radius: 20px;
}
#logoHTML5 {
height: 50px;
width: 70px;
}
#tips {
margin: 0;
padding: 0;
/* float: right;*/
font-size: .6em;
}
#tips ul {
list-style-position: inside;
list-style-type: circle;
margin: 0;
padding: 0 0 0 10px;
}
#redirect {
position: fixed;
text-align: center;
border-radius: 30px;
box-shadow: 0px 0px 150px 150px rgba(0,0,0,0.7);
padding: 30px;
top: 100px;
left: 100px;
right: 100px;
bottom: 100px;
z-index: 100;
font-size: 1.3em;
font-weight: bold;
display: none;
}
#noscript {
position: fixed;
min-height: 20px;
text-align: center;
box-shadow: 0px 0px 15px 15px rgba(0,0,0,0.7);
padding: 5px;
top: 0;
left: 0;
right: 0;
z-index: 100;
font-size: 1em;
font-weight: bold;

}
```

LODE.js

```
var Lode={
Control:  {}
};
/*
* Resizer Controller.  manage the reside slider and the two resizer buttons
*/
Lode.Control.Resizer= new Class({
options:  {
minStep:  120,
maxStep:  680,
sliderLenght:  620,
knobLenght:  60,
margin:  1,
width:  800,
skip:  10
},
//implements the option interface of mootools
Implements:  [Options],
initialize:  function(slider,videoButton,slideButton,video,slide,options){
//set the optional options
this.setOptions(options);
this.videoButton= $(videoButton);
this.slideButton= $(slideButton);
this.slider= slider;
this.video= video;
this.slide= slide;
// add the key control shortcuts
document.addEvents({
'keydown:keys(shift+right)':  function(e) {
this.enlargeVideo();
}.bind(this),
'keydown:keys(shift+left)':  function(e) {
this.enlargeSlide();
}.bind(this)
})
// bind the click event on the buttons with the resize functions
// of this controller
this.videoButton.addEvents({
'click':  function(event) {
this.setVideoBig();
}.bind(this)
}),
this.slideButton.addEvents({
'click':  function(event) {
this.setSlideBig();
}.bind(this)
}),
// on change of the slider, update the slides and the video
this.slider.addEvents({
'change':  function(pos){
this.updateSlide(pos);
this.updateVideo(pos);
}.bind(this)
})
},
setVideoBig:  function(){
new Fx.Tween(this.slide).start('width',this.options.minStep-this.options.margin);
new Fx.Tween(this.video).start('width',this.options.maxStep-this.options.margin);
new Fx.Tween(this.slider.knob).start(this.slider.property,this.options.sliderLenght-this.options.knobLenght)
},
setSlideBig:  function(){
```

```
new Fx.Tween(this.video).start('width',this.options.minStep-this.options.margin);
new Fx.Tween(this.slide).start('width',this.options.maxStep-this.options.margin);
new Fx.Tween(this.slider.knob).start(this.slider.property,0)
},
updateSlide:  function(pos){
this.slide.setStyle('width', this.options.width-this.options.margin-pos)
},
updateVideo:  function(pos){
this.video.setStyle('width', pos-this.options.margin)
},
enlargeVideo:  function(){
this.slider.set(this.slider.step+this.options.skip);
},
enlargeSlide:  function(){
this.slider.set(this.slider.step-this.options.skip);
}
});
/*
* Video Controller.  manage the seek bar, the slides link, the volume and
* playrate sliders and the play/pause ff and rew buttons
*/
Lode.Control.Seeker= new Class({
options:  {
seekRatio:  0,
bufferRatio:  0,
bufferMaxWidth:  800,
activeSlide:  0,
skip:  30
// knobLenght:  15
},
//implements the option interface of mootools
Implements:  [Options],
initialize:  function(seekSlider,volumeSlider,playrateSlider,playPause,rew,ff,
video,slide,slides,buffered,currentTimeLabel,
totalTimeLabel,dataXml,options){
//set the optional options
this.setOptions(options);
this.options.seekRatio= seekSlider.steps/dataXml.data.videoEndTime;
this.options.bufferRatio= this.options.bufferMaxWidth/dataXml.data.videoEndTime;
this.slides= slides;
this.video= video;
this.slide= slide;
this.seekSlider= seekSlider;
this.volumeSlider= volumeSlider;
this.playrateSlider= playrateSlider;
this.playPause= playPause;
this.rew= rew;
this.ff= ff;
this.dataXml=dataXml;
this.buffered=buffered;
this.currentTimeLabel=currentTimeLabel;
this.totalTimeLabel=totalTimeLabel;
// this.playRate= null;
//disable the brwoser controls and set the first slide as active
this.video.controls=false;
this.setActiveSlide(0);
// add the key control shortcuts
document.addEvents({
'keydown:keys(control+right)':  function(e) {
this.ff.fireEvent('mousedown');
}.bind(this),
'keydown:keys(control+left)':  function(e) {
this.rew.fireEvent('mousedown');
}.bind(this),
'keydown:keys(control+up)':  function(e) {
```

```
this.slideGoToPrevious();
}.bind(this),
'keydown:keys(control+down)': function(e) {
this.slideGoToNext();
}.bind(this),
'keydown:keys(space)': function(e) {
this.playPause.fireEvent('click');
}.bind(this)
})
// bind the click on playpause
this.playPause.addEvents({
'click': function(event) {
if (this.video.paused) this.video.play()
else this.video.pause()
}.bind(this)
});
// bind the click on ff
this.ff.addEvents({
'mousedown': function(event) {
this.videoGoToTime(this.video.currentTime+this.options.skip);
this.seekerGoToPos((this.video.currentTime+this.options.skip)*this.options.SeekRatio);
this.checkSlideChange(this.video.currentTime+this.options.skip);
}.bind(this) }),
// bind the click on rew
this.rew.addEvents({
'mousedown': function(event) {
this.videoGoToTime(this.video.currentTime-this.options.skip);
this.seekerGoToPos((this.video.currentTime-this.options.skip)*this.options.SeekRatio);
this.checkSlideChange(this.video.currentTime-this.options.skip);
}.bind(this)
});
//this.video.addEvent("timeUpdate", function (){console.log(this.video.currentTime)});
// add the event on time update
this.video.addEvents({
'timeupdate': function(event) {
this.seekerGoToPos(this.video.currentTime*this.options.seekRatio);
this.checkNextSlide(this.video.currentTime);
this.updateTime(this.video.currentTime);
}.bind(this),
'progress': function(event) {
new Fx.Tween(this.buffered).start('margin-left',this.video.buffered.start()*this.options.bufferRatio)
new Fx.Tween(this.buffered).start('width',this.video.buffered.end()*this.options.bufferRatio)
}.bind(this),
'loadedmetadata': function(event) {
this.totalTimeLabel.set('html',(this.secondsToTime(this.video.duration)));
}.bind(this)
});
// add the click event on the slides
this.dataXml.data.slides.each(function(s,i){
$(i.toString()).addEvents({
'click': function(event) {
this.setActiveSlide(i);
this.seekerGoToTime(s.tempo);
this.videoGoToTime(s.tempo);
}.bind(this)
})
}.bind(this));
// activate the seek bar slider
this.seekSlider.addEvents({
'onChange': function(time) {
// need to try to chain all of these shit!
var c= new Chain();
c.chain(
this.checkSlideChange(time)
).chain(
```

```
this.updateTime(time)
).chain(
this.videoGoToTime(time)
)
}.bind(this)
});
//activate the volume and playbackrate sliders
this.volumeSlider.addEvents({
'onChange': function(v) {
v= (v*0.1).toFixed(1);
this.setVolume(v);
this.volumeSlider.knob.set('html',v)
}.bind(this)
});
this.playrateSlider.addEvents({
'onChange': function(v) {
v= (1+v*0.2).toFixed(1);
this.setPlaybackRate(v);
this.playrateSlider.knob.set('html',v)
}.bind(this)
});
},
updateTime: function(s) {
this.currentTimeLabel.set('html',this.secondsToTime(s));
},
secondsToTime: function(s) {
var h=Math.floor(s/3600);
var m=Math.floor((s/60)%60);
var c=Math.floor(s%60);
m= (m<10)? '0'+m : m;
c= (c<10)? '0'+c : c;
return h+":"+m+":"+c;
},
videoGoToTime: function(time){
time= (time<0) ? 0 : time;
time= (time>this.video.totalTime0) ? this.video.totalTime : time;
this.video.currentTime= time;
},
seekerGoToPos: function(pos){
this.seekSlider.knob.setStyle(this.seekSlider.property, pos);
},
seekerGoToTime: function(time){
time= (time<0) ? 0 : time;
time= (time>this.video.totalTime0) ? this.video.totalTime : time;
this.seekSlider.knob.setStyle(this.seekSlider.property, time*this.options.seekerRatio);
},
checkSlideChange: function (time) {
time= (time<0) ? 0 : time;
time= (time>this.video.totalTime0) ? this.video.totalTime : time;
var i;
if( time<this.dataXml.data.slides[this.options.activeSlide].tempo) {
i=-1;
while( this.dataXml.data.slides[++i].tempo < time) {}
this.setActiveSlide(i-1);
} else{
if ( typeof this.dataXml.data.slides[this.options.activeSlide+1] === 'undefined') return;
if(time>this.dataXml.data.slides[this.options.activeSlide+1].tempo) {
i=this.dataXml.data.slides.length;
while( this.dataXml.data.slides[--i].tempo > time) {
}
this.setActiveSlide(i);
}
}
},
checkNextSlide: function (time) {
```

```
    if ( typeof this.dataXml.data.slides[this.options.activeSlide+1] === 'undefined') return;
    if(time>this.dataXml.data.slides[this.options.activeSlide+1].tempo) {
    this.setActiveSlide(this.options.activeSlide+1);
    }
    },
    setActiveSlide:  function(id) {
    $(this.options.activeSlide.toString()).removeClass('activeSlide');
    this.options.activeSlide= id;
    $(this.options.activeSlide.toString()).addClass('activeSlide');
    this.slide.set('src',this.dataXml.data.slides[id].immagine);
    $('slideLink').set('href',this.dataXml.data.slides[id].immagine);
    },
    slideGoToNext:  function() {
    if (this.options.activeSlide >= this.dataXml.data.slides.length-1) return;
    this.setActiveSlide(this.options.activeSlide + 1);
    this.videoGoToTime(this.dataXml.data.slides[this.options.activeSlide].tempo);
    this.seekerGoToTime(this.dataXml.data.slides[this.options.activeSlide].tempo);
    this.updateTime(this.dataXml.data.slides[this.options.activeSlide].tempo);
    },
    slideGoToPrevious:  function() {
    if (this.options.activeSlide==0) return;
    this.setActiveSlide(this.options.activeSlide - 1);
    this.videoGoToTime(this.dataXml.data.slides[this.options.activeSlide].tempo);
    this.seekerGoToTime(this.dataXml.data.slides[this.options.activeSlide].tempo);
    this.updateTime(this.dataXml.data.slides[this.options.activeSlide].tempo);
    },
    setVolume:  function(v) {
    this.video.volume= v;
    },
    setPlaybackRate:  function(v) {
    this.video.playbackRate= v;
    }
    });
    /*
    * Loader of the xml data
    */
    Lode.XmlDataLoader= new Class({
    data:  {
    },
    //implements the Chain interface of mootools
    Implements:  Chain,
    initialize:  function (rawXml){
    this.parse(rawXml);
    },
    parse:  function (rawXml){
    //read the video information
    var f=rawXml.getElementsByTagName('video')[0];
    this.data.videoName= f.getElementsByTagName('nome')[0].childNodes[0].nodeValue;
    this.data.videoEndTime= f.getElementsByTagName('totaltime')[0].childNodes[0].nodeValue;
    this.data.videoStartTime= f.getElementsByTagName('starttime')[0].childNodes[0].nodeValue;
    // load the lecture info
    f=rawXml.getElementsByTagName('info')[0];
    this.data.corso= f.getElementsByTagName('corso')[0].childNodes[0].nodeValue;
    this.data.titolo= f.getElementsByTagName('titolo')[0].childNodes[0].nodeValue;
    this.data.professore= f.getElementsByTagName('professore')[0].childNodes[0].nodeValue;
    this.data.dinamicUrl= f.getElementsByTagName('dinamic_url')[0].childNodes[0].nodeValue;
    //load all the slide elements into an array
    var slides= rawXml.getElementsByTagName('slide');
    //create an array of the right length in the data JSON
    this.data.slides=[];
    //for every slide i load the data and create a JSON object and inject it into the data JSON slide
array
    Array.each(slides, function(slide, index){
    //need to create the JSON before insert data into it
    this.data.slides[index]={};
```

```
this.data.slides[index].tempo=Math.floor((slide.getElementsByTagName('tempo')[0].childNodes[0].nodeValue).toInt());
this.data.slides[index].titolo=slide.getElementsByTagName('titolo')[0].childNodes[0].nodeValue;
this.data.slides[index].immagine=slide.getElementsByTagName('immagine')[0].childNodes[0].nodeValue;
}.bind(this));
}

});
```

init.js

```
document.addEvent('domready',function(){
// retrive all the object required by the classes
var video= $('video');
var slide= $('slide');
var slides= $('slides');
var bigVideo= $('bigVideo');
var bigSlide= $('bigSlide');
var playpause=$('playPause');
var rew=$('rew');
var ff=$('ff');
var buffered=$('buffered');
var currentTimeLabel=$('currentTimeLabel');
var totalTimeLabel=$('totalTimeLabel');
// load the raw data in the XML file
var dataXml= "";
new Request({
url:  'data.xml',
method:  'post',
onSuccess:  function(){
my=new Chain();
my.chain(
dataXml= new Lode.XmlDataLoader(this.response.xml)
).chain(
xmlDataSetUp()
).chain(
videoControlOn()
)
},
onError:  function(){console.log('Error Loading the XML data');}
}).send();
//#### RESIZE CONTROLS INITIALIZATION
// create the resize slider
var resizeSlider = new Slider($('resizeContainer'), $('resizeHandle'), {
steps:  560,
range:[120,680],
initialStep:  398
});
//create the resize controller
new Lode.Control.Resizer(resizeSlider,bigVideo,bigSlide,video,slide);
//#### SEEK CONTROLS INITIALIZATION
// a function cause it's called as a chain after loading the xml
function videoControlOn() {
// create the sliders controllers
var seekSlider = new Slider($('seekContainer'), $('seekHandle'), {
steps:  785,
range:[dataXml.data.videoStartTime,dataXml.data.videoEndTime],
initialStep:  dataXml.data.videoStartTime
});
var volumeSlider = new Slider($('volumeContainer'), $('volumeHandle'), {
steps:  11,
range:  [0, 10],
snap:  true,
initialStep:  10
});
var playrateSlider = new Slider($('playrateContainer'), $('playrateHandle'), {
steps:  11,
range:  [0, 10],
snap:  true,
initialStep:  0
});
//finally create the video controller
```

```
new Lode.Control.Seeker(seekSlider,volumeSlider,playrateSlider,playpause,rew,ff,
video,slide,slides,buffered,currentTimeLabel,totalTimeLabel,dataXml);
}
function xmlDataSetUp() {
$('titolo').set('html',dataXml.data.titolo);
$('docente').set('html',dataXml.data.professore);
$('corso').set('html',dataXml.data.corso);
//reset the text "loading slides"
slides.set('html','');
//evaluate the ratio for the width
var ratio=(785/dataXml.data.videoEndTime);
//for each slide, create a li element and a div for the seek bar
dataXml.data.slides.each(function(slide,index){
//if the first one
if (typeof dataXml.data.slides[index+1] === 'undefined') {
width=Math.round((dataXml.data.videoEndTime-slide.tempo)*ratio);
} else {
width=Math.round((dataXml.data.slides[index+1].tempo-slide.tempo)*ratio);
}
//remove the border size
width-=2;
// create a new 'li' element
var slideElement = new Element('li', {
id :  index,
html:  slide.titolo
});
var barElement = new Element('div', {
id :  'div'+index,
'class' :  'seekSlide',
style :  'width:'+width+'px',
title :  slide.titolo
});
//
// inject the elements in the page
slideElement.inject(slides);
barElement.inject($('seekSlides'));
})
}
})
function fallback() {
$('redirect').setStyle('display','block');
(function(){location.href="start.html"}).delay(3000);

}
```

Riferimenti bibliografici

[W3C] http://www.w3.org/TR/html5/

[RDP] M. Ronchetti M. Dolzani, Lectures on demand: the architecture of a system for delivering traditional lectures over the web., Kommers, P., & Richards, G. (Eds.), Conference on Educational Multimedia, Hypermedia and Telecommunications 2005, 2005, pp. 1702-1709