MediaElements, TrackElements & MediaControllers

Enabling Multi-device, Web-based, Linear Media with Shared Motion

Ingar Arntzen*, Njål Borch*, François Daoust**, Dominique Hazael-Massieux**
* Norut and Motion Corporation, ** World Wide Web Consortium

1. Introduction

On the Web, development of linear media is supported by three main concepts; HTMLMediaElement, HTMLMediaController and HTMLTrackElement. In combination, these concepts enable a powerful programming model for web-based, linear media. In particular, web-based, linear media may leverage core features of the Web platform, such as programmability, interactivity and access to online resources and services. Such features, as well as the universal availability of the platform, make the Web an ideal foundation for linear media.

However, the current model also suffers from a crucial limitation. *Media elements*, *tracks* and *controllers* are inherently single-device, making the development of *multi-device*, linear media challenging at best. At the same time, the importance of multi-device web applications is ever increasing. This development is fueled by the abundance of smart devices, as well as the need to support seamless workflows and presentations across devices. In particular, companion device applications for broadcast TV receives much attention in the industry. Many such applications would benefit from precise linear coordination (along the program timeline). Looking ahead, we envision a programming model for web-based linear media that is multi-device by default, turning single-device linear media into a special case.

In this paper we revise the current programming model for web-based, single-device, linear media, and explain how it can be transformed into a programming model for multi-device, linear media. Shared Motion [MSV] is the key concept enabling this transformation. We propose adding a new, simplified media controller to HTML5, and follow up with minor adjustments to media elements and track elements. These changes, we argue, would unlock a powerful programming model for multi-device web applications.

MediaScape

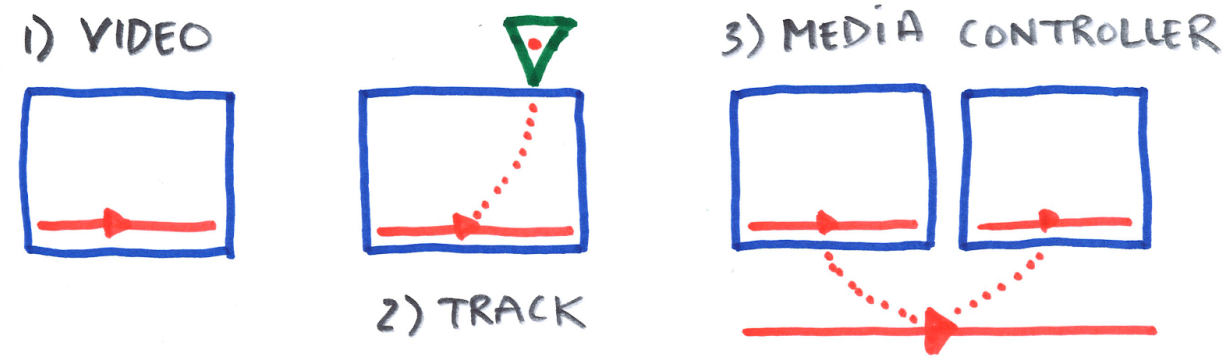
This research is financed in part by MediaScape (EU, FP7-ICT-2013-10, grant agreement 610404) http://mediascapeproject.eu/. The ideas presented in this paper has impacted the architectural approach to multi-device Web applications currently explored by MediaScape.

Shared Motion

Shared Motion is a novel concept/approach for multi-device media control and synchronization, and a central aspect of this paper. The concept is defined by Ingar Arntzen and Njål Borch, working as researchers at Norut. Motion Corporation, founded by Arntzen and Borch, has implemented an online service *InMotion* based on this concept. Properties claimed for Shared Motion as a concept is backed by measurements and experience with development using the InMotion service. Client side libraries are open source, and the InMotion service is publicly available. Shared Motion is explored as basis for multi-device timing and synchronization in MediaScape.

2. Web-based Linear Media

The below figure shows three familiar cases in web-based linear media, and at the same time indicates the roles of the three main concepts; media elements, tracks and controllers.



1) **HTMLMediaElements** are self contained UI components encapsulating presentation of continuous media (blue), as well as associated media control and progress (red). In this paper the term media element refers to implementation of the HTTPMediaElement API, i.e., HTML5 Video/Audio.

2) **HTMLTrackElements** (green) provide a mechanism for aligning timed data with the progress of a media element. Tracks work on cues (JSON, WebVTT, etc.) associated with time intervals and provide timely event upcalls (i.e., enter & exit) during media playback. For instance, tracks allow linear media to be enriched by metadata such as subtitles and chapter information. More generally, tracks provide a mechanism for transforming timed data (JSON) into linear presentation, or timed actions (JavaScript) into timed execution.

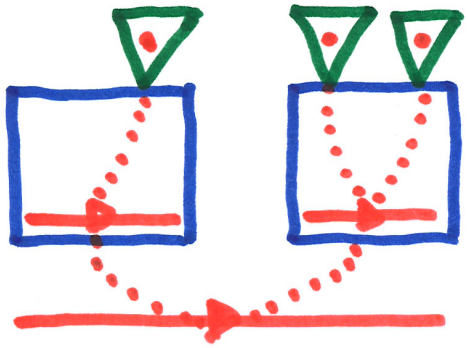
3) **HTMLMediaControllers** (red) provide a mechanism for co-presenting multiple media elements (and tracks by implication). For example, Media Controllers can be used to orchestrate consistent presentation of multiple camera angles, and provide media controls that affect all media elements in unison.

Though specific to the Web, these three concepts represent aspects that are common to many media frameworks. In essence, media elements target presentation of *continuous* media, whereas tracks provide a helpful mechanism for presenting timed, *non-continuous* media. Finally, media controllers (as the name suggests) encapsulate control and coordination of multiple independent linear media sources.

Current frameworks for linear media, be it SMIL, Silverlight, Flash, Popcornjs [SMIL, SILV, FLA, POP] all have similar concepts. Still, in some cases similar concepts are not represented as independent constructs, but entangled and integral to a closed or monolithic frameworks. In contrast, the way of the Web is opening up, providing programmability and extensibility by flexible combination of simpler components. In the next section we explore how media elements, track elements and media controllers support such linear composition.

3. Linear Composition

The distinction between media elements, tracks and controllers emphasizes that composition as a design principle extends naturally to web-based, linear media. Advanced media can be constructed from simpler components, essentially by combining these concepts in a variety of ways. *Linear composition* implies that media elements and track elements all behave consistently, with reference to a single, shared media controller.



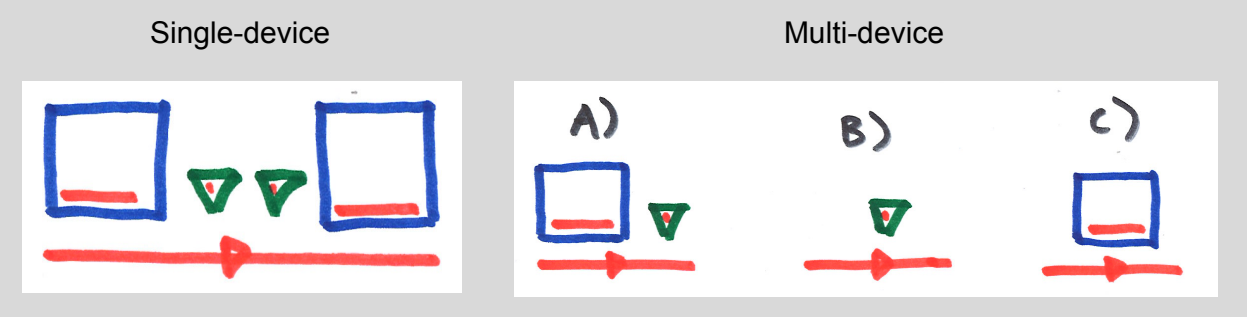
In the above figure one media controller, two media elements and three track elements have been combined to form a single, coordinated presentation. Track elements depend on media elements (they are children in the DOM).

Regatta Example

For example, a regatta could be presented from a collection of timed media, e.g. official video feed, timing info, GPS coordinates and commentary, as well as crowd-sourced video-clips, pictures and Twitter messages. The entire event would then be replayed and navigated using the media controller as common timeline. Video-clips would be aligned using media elements while track elements control map displays, images and text.

Single-device to Multi-device

The concept of linear composability extends naturally to the multi-device scenario. Essentially, we want to go from single-device to multi-device by scattering or duplicating components across devices. At the same time, we go from single-device playback, to simultaneous, multi-device playback.



The above figure illustrates how two media elements and two track elements may be split across three devices. In order to support multi-device linear composition, the challenge is to ensure that the three media controllers in a), b) and c) are kept in synchrony. This is the topic of the next section.

Note also that the current dependency between track elements and media elements is not appropriate for the multi-device scenario. In this illustration, track elements are promoted as standalone programming constructs depending directly on media controllers, just like media elements do.

Regatta Example Revisited

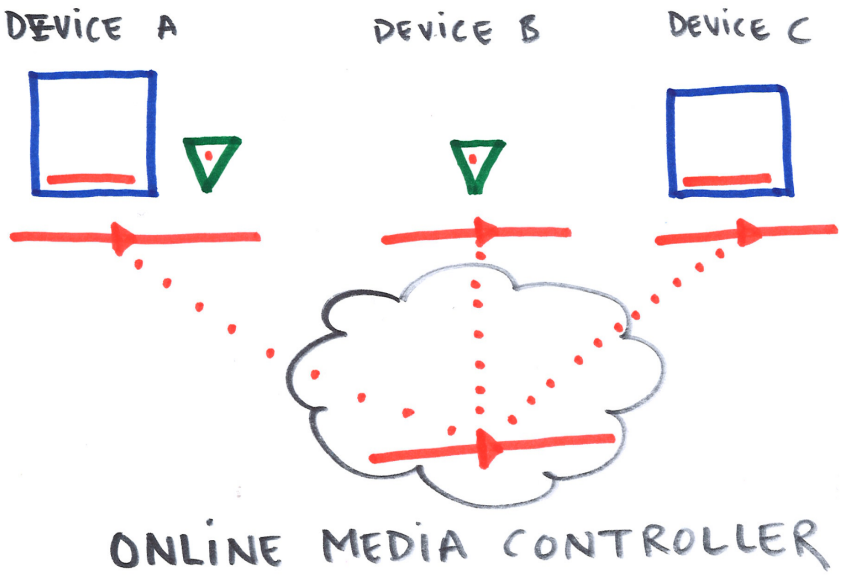
Going back to the regatta example, interactive race infographics and the regatta map may be hosted by an IPad, while a smart TV presents the main video feed. A smart phone may present time-aligned video-clips, photos and comments, while at the same time serving as an input-device for same user generated content. Finally, media control is available from all devices. For instance, a touch-sensitive regatta timeline on the IPad may support easy timeshifting, as would a simple progress bar on the smart phone. Media control affects all components in unison, thereby providing consistent linear composition across multiple devices.

4. Shared Media Control

From the previous section it should be clear that the main technical challenge with multi-device, linear composability, is multi-device media control. In the single-device scenario, all components are connected to a single, shared media controller. We propose to extend this notion to the multi-device domain as well.

In multi-device, linear media, distributed media elements and track elements are equally connected to a single, shared media controller.

Furthermore, since we are designing for the Web, shared media controllers should be available wherever the Web is available. It follows that technical solutions based on services or features of local networks, or specific network carriers, are not appropriate. Also, in line with the client-server architecture of the Web, we prefer a centralised, service-based solution. So, we propose the concept of *online* media controllers, hosted by web services and available for all connected devices.



The above figure illustrates a single, online media controller, shared between media elements and track elements, distributed across three connected devices. The media controller on each device serves as *proxy* for the shared, online media controller. By emulating the interface of current media controllers, media elements and track elements may readily support linear composition in multi-device as well as single-device media.

The concept of online media controllers should integrate well with the existing model. In the terminology of the DOM we are essentially adding a source attribute to the current media

controller. This way, devices providing the same source URL for their media controllers will automatically experience symmetric control and coordinated playback.

Though an attractive prospect, the idea of shared, online media controllers raises some questions; How can shared media controllers be implemented, and how well can they be synchronised? Fortunately, a solution to this problem already exists. Shared Motion [MSV] is a generic mechanism for web-based, multi-device media control, and the topic of the next section.

5. Shared Motion

HTMLMediaControllers emulate the full API of HTMLMediaElements, including aspects such as buffering and volume control. However, the essential property needed for synchronisation is simply the media offset, or more precisely, how the media offset changes in time. In other words, **media controllers manage linear *motion* through linear media**. The concept of *Shared Motion* encapsulates this perfectly, and is designed specifically to allow precise and highly scalable media control and synchronization across the Internet. Motion Corporation has implemented *InMotion*, an online service making Shared Motion available for the Web. Shared Motion works in any modern Web-browser, no plugins required. Still, even though Shared Motion is designed for the Web it is not restricted to the Web. As such, Shared Motion provides a mechanism for distributed linear composition, applicable to all connected agents, from web-browsers on laptop computers to native applications on embedded devices.

A general timing mechanism for the Web

In a broader perspective, Shared Motion is essentially a distributed timing mechanism for the Web. As such it is useful for a variety of timing-related challenges in multi-device applications. In particular, Shared Motion is basis for distributed media control, remote control, synchronization, simultaneity, time-shifting, time-sensitive recording, and time-sensitive linear composition, all in one simple concept.

MIDI Analogy

The timing aspects of the MIDI (Musical Instrument Digital Interface) protocol [MIDI] may serve as a helpful analogy to Shared Motion. In the music industry, MIDI is widely used (through three decades) as a timing mechanism for coordinating and synchronising instruments and devices in musical production. For instance, MIDI may control playback from synthesizers, samplers, drum machines, computers, or even stage lighting and events in theatrical productions. Web MIDI API [WEBMIDI] is a recent W3C standardization effort allowing web browsers direct access to MIDI devices, or to synchronize with external MIDI controllers.

However, crucially, the scope of MIDI is limited to geographical vicinity. This is because the underlying synchronisation protocol depends on latency of electrical pulses over physical

cables (or wireless links). Shared Motion is analogous to a MIDI clock, with the significant exception that Shared Motion works globally. This implies that Shared Motion may be used to emulate global MIDI, and that browser support for Shared Motion would boost current developments such as WebMIDI and WebAudio [WEBAUDIO]. More importantly, it means interoperability through precise time-sensitive coordination becomes available for the Web at large.

Media Control as a Service

The idea of online media controllers naturally entails the idea of specialized hosting services. InMotion by MotionCorporation is an example of this. *Media Control as a Service* implies potential cost reduction, high availability, reliability, precision, scale, and (not least) simplicity of use. These are all critical factors for enabling multi-device, timing-sensitive Web applications.

Note also that Media Control as a Service implies content independence. For example, InMotion exclusively hosts *motions* (i.e., synchronization points), and does not participate in media distribution. This model implies a two-step approach to media synchronisation:

1. Shared Motion is synchronized between a hosted service and connected clients.
2. Timed media content is then synchronized locally at client-side, relative to motion proxies.

This approach is beneficial for a number of reasons.

- Content transport does not have to go via a third party synchronization services, but may go directly to consumers. The third party motion provider can remain agnostic concerning media content and purposes for synchronization.
- Shared Motion is extremely light-weight, and the precision of cross-Internet motion synchronization is equal to application level clock synchronization [MSV, SYNQ].
- Client-side media synchronization can be performed by standard media elements and track elements.
- Client-side media synchronization may potentially mask the effects of network delay for transfer of media content.
- Content-independent synchronization is open to different approaches to media dissemination, e.g., pull-based or push-based.

Shared Motion (as implemented by InMotion) can be expected to synchronize with expected errors < 10 ms in modern web-browsers [SYNQ] under most network conditions. Shared Motion is pure web technology, and as such it works independent of network carrier, OS or browser type. It may provide excellent synchronization even in circumstances with poor network connection (e.g. edge), yet a modest reduction in precision is to be expected. Furthermore, Shared Motion synchronizes very quickly, within fractions of a second after connecting.

The implementation of Shared Motion used by Motion Corporations InMotion service is based on the technical concept of *Media State Vectors* [MSV]. MSV's represent distributed, deterministic motion and are based on the classical equations for linear motion under uniform acceleration. For this reason, Shared Motions are controlled by specifying position, velocity and/or acceleration. This way, Shared Motion is highly expressive and supports a variety of control primitives appropriate for different kinds of media. For example, Shared Motion supports play/pause/fast-forward (continuous media, time lapses, etc), or next/previous/goto (slideshows, image galleries, etc).

6. Proposal

The prospect of Web support for multi-device, linear composability lead us to the following proposal.

1. Introduce an ideal media clock as new type of media controller in HTML5, and extend HTMLMediaElements and HTMLTrackElements with appropriate support.
2. A common interface for online media control.

The first proposal is the important one. A new type of media controller, along with certain improvements to media elements and track elements, would significantly improve Web support for multi-device, linear composability. The second proposal would be something to consider if the first one was implemented.

7. A new type of media controller

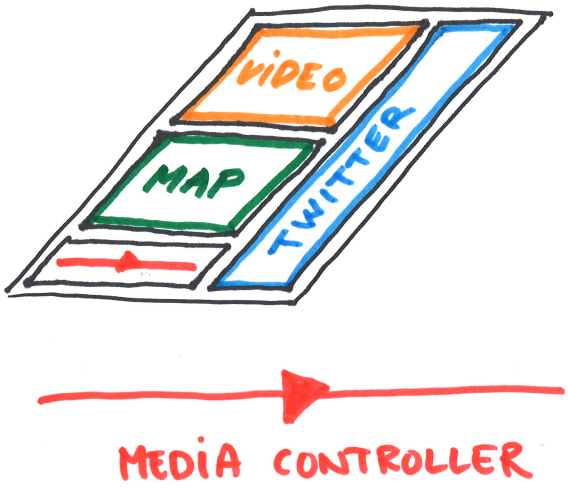
The HTMLMediaController currently emulates the full API of MediaElements. This includes concepts of readiness, buffering and volume.

For the purpose of precise linear composition in both single-device and multi-device media, we propose to extend HTML5 with a new, simplified media controller implementing an ideal media clock.

Also, in the interest of backward compatibility, the current HTML Media Controller could additionally be extended to accept external directions from this simplified media controller.

A media controller working as an ideal media clock would imply that the media controller has no concept of readiness or buffering. So, if a media element needs to buffer, the media controller would simply go on undisturbed. This might break with conventions, but we argue that it is correct with respect to linear composition. After all, halting a full experience simply because one of the video feeds are struggling is not always what you want. This is even more true in multi-device scenarios involving multiple devices and people. On the other hand, if the correct behavior is indeed to halt the progress, this can easily be accomplished by halting the media controller manually by user input, or halting it from JavaScript upon detecting buffering issues.

Additionally, a simplified media controller would be applicable to other types of linear media, besides media elements. For instance, the Web Animation Framework [ANI] is based on the concept of media clocks. It should be easy to create a single presentation by combining Web animations with media elements and track elements.



The above figure illustrates how different linear UI components may be directed by a single, shared media controller, in a single-device presentation. A simplified media controller may ease integration with custom components, by not requiring them to emulate the full API of media elements. For instance, a custom component could be a third party widget for time-aligned Twitter messages.

The expressiveness and general nature of Shared Motion (i.e., the Media State Vector) makes it a candidate implementation for such an ideal media clock.

Adjustments to media elements and track elements

Complementing the simplified media controller, it would also be useful to improve aspects of media elements and track elements. For example, in order to synchronise with appropriate precision between media elements (e.g., different camera angles or alternative audio tracks to video), media elements must be optimized for external control. In particular, media elements should always compensate for internal delays (buffering etc.) by adjusting their internal playback offset according to the external media controller. In addition, to maintain synchrony, it is necessary for media elements to compensate for playback drift relative to the media controller.

The track element too would benefit from improvement with respect to precision. Our initial tests indicate that cue events are emitted rather coarsely. In Chrome version 39 for example, the events appear to fire about 150-250 milliseconds too late according to the video currentTime. Furthermore, it is important that the track element provides events not only during playback, but also in response to seekTo operations, or dynamic changes in the tracks content (i.e., cues).

If implemented, this proposal would imply that media elements and track elements are readily available for precise and flexible linear composition in the single device scenario. Furthermore, with simple media controllers being ideal media clocks it would be easy to integrate with solutions for online media control. As a result, the Web would greatly improve its support for multi-device linear composition.

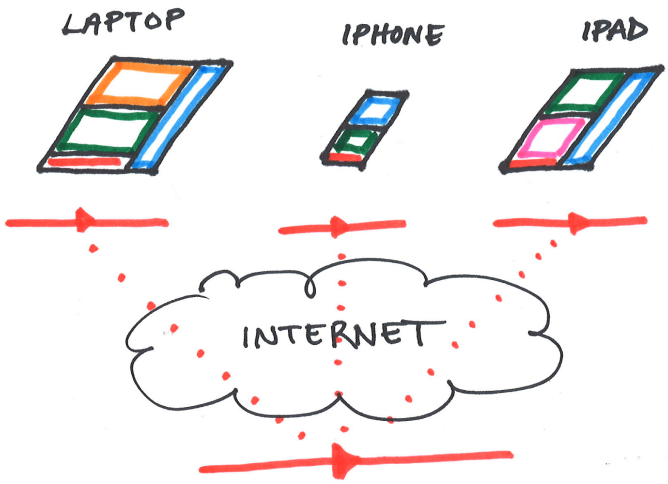
We have verified the feasibility of these proposed changes by implementing these changes in JavaScript. The JavaScript wrapper for synchronization of media elements already demonstrates sub-framerate precision in multi-device synchronization [SYNQ]. However, media elements behave differently with different browsers, different media and on different architectures. It is hard to maintain a library for all permutations. Even worse, properties affecting synchronization are sensitive to change across browsers updates. It would be much better if these problems were addressed by media elements internally.

We have also implemented an improved version of the track element which reliably delivers event upcall at the correct millisecond. The effect of this is most notable in multi-device

scenarios or if multiple pieces of content are tightly bound (e.g. flash a border in sync with an explosion in a video). Furthermore, our implementation guarantees enter and exit events to always be consistent with any kind of media control supported by Shared Motion. This even includes acceleration. Consistency of enter and exit events additionally extends to dynamic changes to cues. For instance, this implies that a timed subtitle may safely be requested to shrink, shift or be prolonged in time, dynamically during playback, without added complexity for the programmer.

8. A common interface for online media control

In the interest of interoperability, we would suggest the possibility of defining a common interface between providers of online media control and web browsers. This would allow linear composability to apply for all things linear, including media elements, proprietary media frameworks or custom web components. In other words, it would enable web-based mash-up to extend naturally to linear media.



The above figure illustrates how the media controller acts as a mediator between an online motion provider and independent components in a linear presentation. By ensuring that the local media controller and the interface to online media control adopt a common API, complexity of multi-device linear composition is further reduced. We currently do this for Shared Motion, as proxies to online motions are indistinguishable from local motions. This allows linear UI components to be developed towards a single API and play part in single-device linear media as well as multi-device media, without modification.

We suggest Shared Motion as a candidate technology for such a common interface.

The interface definition essentially consist of two parts, the programmer API for the media controller, and the protocol for communication with motion providers. The protocol, as currently implemented by the InMotion service, is strictly client-server, involving JSON messages communicated across a single WebSocket connection. This solves both matters of clock synchronization as well as dissemination of motion updates (e.g. play/pause commands or more technically, media state vectors). InMotion additionally supports fallback to regular HTTP-based communication, in case WebSockets are not supported.

We believe it will be possible to define a common interface based on this approach, and that this protocol would fit well into the family of Web protocols managed by the W3C.

References

[MIDI] Musical Instrument Digital Interface http://en.wikipedia.org/wiki/MIDI

[WEBMIDI] Web MIDI API http://webaudio.github.io/web-midi-api/

[WEBAUDIO] Web Audio API
https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html

[MCORP] Motion Corporation motioncorporation.com

[MSV] *Ingar M. Arntzen, Njål T. Borch and Christopher P. Needham. 2013. "The Media State Vector: A unifying concept for multi-device media navigation". In Proceedings of the 5th Workshop on Mobile Video (MoVid '13). ACM, New York, NY, USA, 61-66.*

[SYNQ] Njål T. Borch and Ingar M. Arntzen, “Distributed Synchronization of HTML5 Media” (15/2014), http://norut.no/nb/node/3041

[SMIL] Synchronized Multimedia Integration Language
http://www.w3.org/TR/SMIL3/smil-timing.html

[SILV] Microsoft Silverlight http://www.microsoft.com/silverlight/

[FLA] Adobe Flash http://www.adobe.com/products/flashplayer.html

[POP] Popcornjs The HTML5 Media Framework http://popcornjs.org/

[ANI] Web Animations http://www.w3.org/TR/web-animations/