Multi-device Linear Composition on the Web

Enabling Multi-device Linear Media with HTMLTimingObject and Shared Motion

Ingar M. Arntzen
Njål T. Borch
Northern Research Institute
Tromsø, Norway
ingar@norut.no
njaal@norut.no

François Daoust
Dominique Hazael-Massieux
World Wide Web Consortium
Paris, France
fd@w3.org
dom@w3.org

Abstract

Composition is a hallmark of the Web, yet it does not fully extend to linear media. This paper defines linear composition as the ability to form linear media by coordinated playback of independent linear components. We argue that native Web support for linear composition is a key enabler for Web-based multi-device linear media, and that precise multi-device timing is the main technical challenge. This paper proposes the introduction of an HTMLTimingObject as basis for linear composition in the single-device scenario. Linear composition in the multi-device scenario is ensured as HTMLTimingObjects may integrate with Shared Motion, a generic timing mechanism for the Web. By connecting HTMLMediaElements and HTMLTrackElements with a multi-device timing mechanism, a powerful programming model for multi-device linear media is unlocked.

Author Keywords

linear composition; temporal composition; interoperability; multi-device; distributed; timing; synchronization; media control; Web; HTMLTimingObject, HTMLMediaElement, HTMLTrackElement.

ACM Classification Keywords

H.5.1 [Multimedia Information Systems]: Linear Media; H.5.4 [Hypertext/Hypermedia]: Web; D.1.3 [Distributed Programming]: Timing, control and synchronization; D.2.12

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: a novel, service-based approach to multi-device timing, media control and synchronization. *Motion Corporation* http://motioncorporation.com/ provides an online service *InMotion* offering multi-device timing for Web clients. Client-side libraries are open source, and the InMotion service is publicly available. Shared Motion is explored as basis for multi-device linear media in MediaScape.

W3C Multi-device Timing Community Group https://www.w3.org/community/webtiming/ aims to extend the Web with native support for multi-device timing, adopting the Shared Motion approach as a starting point.

[Distributed Objects]: Interoperabilty linear components

Introduction

Composition as a design principle is a hallmark of the Web, ensuring reusability, extensibility, flexibility and mashup-ability. As the Web has become a rich platform for linear media, one would expect these benefits of composition to apply equally well for production of linear content. In short, linear composition implies that complex linear presentations may be constructed by combining simpler linear components. For instance, a single Web presentation could be formed by the coordinated playback of video [1, 7, 12], timed meta-data run by popcornjs [9], a timed Twitter widget, a map with timed geolocations, a SMIL [10] presentation, some animations based on Flash [1] or WebAnimation [14], a Prezi [4] slide deck and a time-sensitive ad banner within an iFrame. Linear composition would also enable individual components to be loaded or removed dynamically during playback, either as a feature of the storyline, or as a reaction to user input or changes in execution environment. Imagine for instance a TV program delivered to a smart phone, adapting dynamically to the loss of WiFi by replacing the HD video with light-weight infographics based on timed HTML, while keeping the audio.

Unfortunately, the current support for linear composition is weak. While the Web has wide support for linear media, including both native and external frameworks, aspects of timing and control are largely internal and custom to each framework. For the programmer this means that coordinating all the components and overcoming heterogeneity may quickly become a challenge. Loading and removing linear components dynamically during playback adds further complexity. Under such circumstances, non-functional requirements such as precise and reliable synchronization will require hard work, if at all possible.

Equally important, linear composition extends naturally to multi-device scenarios. For example, companion device scenarios imply coordinated playback of linear components across TV's, laptops, tablets and smart phones. Similarly, collaborative viewing requires coordination of the same linear components across multiple devices. Furthermore, seamless workflows and presentations across multiple devices may require loading and removing of linear components dynamically, during presentation, for instance as devices join and leave. This is multi-device linear composition.

In this paper we address linear composition both in single and multi-device scenarios, and outline how the current programming model of the Web may be transformed to support linear composition. We argue that the key enabler for linear composition on the Web is precise distributed timing as a basis for coordinated playback and control. A solution for this is already available. Shared Motion (i.e., Media State Vector (MSV) [2]) implements distributed multi-device timing and media control for the Web, and provides an excellent basis for linear composition in multi-device media. We are proposing Web support for Shared Motion through a novel HTMLTimingObject. In this paper we particularly discuss integration of the HTMLTimingObject with HTMLMediaElements and HTMLTrackElements, a step towards native support for linear composition on the Web.

Related Work

Linear (temporal) composition has been explored before both in single-device and multi-device scenarios. In the Web context, both SMIL [10] and Flash [1] are frameworks that do linear composition internally. However, this paper targets linear composition in a broader context; between heterogeneous frameworks and a variety of timing-sensitive Web components. The key issue is how coordination of linear components is performed.

Single-device Regatta
Figure 1 may illustrate how 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, images and Twitter messages. The entire event could then be replayed and navigated using the media controller as common timeline. Video-clips would be aligned using media elements while track elements would control map displays, images and text.

A first approach to coordination is to allow general application data to be communicated between components. For instance, a master component may push objects to other components, which then implement appropriate reactions. For instance, SMIL State [5] allows communication through shared variables. Similarly, in the multi-device scenario, media content may be pushed from a TV/STB to companion devices, for instance using the Intranet as communication channel. In such data-driven approaches, interfaces between components tend to be application specific. This may imply complex integration and low flexibility. This is exemplified by [6] where seamless, dynamic video composition requires considerable integration work. Still, this is only a very basic use case for linear composition.

A common improvement is to limit communication between components by exchanging only timing and control messages, not general application data. For example, only the currentTime property of the HTMLMediaElement [12] is used by HTMLTrackElement [13] (or frameworks like popcornjs [9]) in order to co-present timed data with media playback. In the multi-device scenario, video playback on Chromecast may be remote controlled by exchanging small control messages over the Intranet, while the video data is fetched directly from Internet. As interfaces for timing and control can be generalized more efficiently (than application specific data interfaces), this may improve reusability. In addition, this approach is more flexible as components may fetch application data independently.

However, the above approaches have a weakness in common; They both introduce dependencies between components. In short, if a component depends on another component, for either data or timing/control, linear composition quickly gets complicated. This observation inspires our approach. We argue that coordination must be achieved while maintaining strict independency between components. This paper explains how this can be done in the Web, both for single-device and multi-device scenarios.

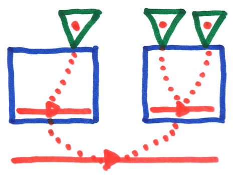
Figure 1: Advanced case for Web-based linear media. Coordinating multiple media elements (blue rectangles) and track elements (green triangles) using a shared media controller.

Linear Composition on the Web
As a starting point for discussing linear composition on the Web, we consider three concepts, central to the Web's native support for linear media; HTMLMediaController, HTMLMediaElements and HTMLTrackElements [11, 12, 13].

In figure 1 one media controller (red line), two media elements (blue rectangles) and three track elements (green triangles) have been combined to form a single, coordinated presentation. Track elements depend on media elements as they are children in the DOM. Media controllers control the playback of the two media elements, as well as the track elements by implication. So, linear composition is already an important feature of the Web.

Linear composition implies that advanced linear media may be constructed by the coordinated playback of independent linear components.

Or, re-formulated in Web terminology:

Linear composition implies that media elements and track elements all behave consistently, with reference to a single, shared media controller.

Unfortunately, there are weaknesses in the current approach. In particular, the scope of the current media controller is very limited, as it is designed exclusively for coordination of media elements. Furthermore, the current media controller bundles timing control with other control aspects, such as buffering and volume control. If one media element requires buffering during playback, the media controller automatically halts all media elements. This might be undesired as default behavior, particularly in multi-device scenarios. For other kinds of linear media, buffering and volume control might not even be relevant. Another weakness is precision in timing control. The media controller depends on the timing model of media elements, essentially a non-deterministic, pulse-based model based on repeated firing of the *timeupdate* event. The lack of resolution and predictability in this model limits precision.

We argue that generic support for linear composition requires a media controller that supports high expressiveness with respect to control primitives, and high precision with respect to timing.

Expressiveness. A generic approach to media control must support control primitives appropriate for various types of linear media and a variety of use cases. For example, slide-show presentations support next, previous or goto whereas continuous media support play, pause and seekTo. Animation frameworks such as SMIL [10] and WebAnimations [14] even support accelerated behaviour.

Precision. A generic approach to media control must allow linear components to be synchronized precisely. For instance, to support linear composition between audio and video, lip-sync precision is required. Or, Internet radio presented by both kitchen and living room devices will produce echo unless sub-framerate precision is supported. A deterministic model for timing is required for precise synchronization. This is especially true in a distributed scenario.

Figure 2: The HTMLTimingObject is essentially an advanced stop-watch.

The HTMLTimingObject

A key issue for linear composition is precise and expressive timing controls. We propose the introduction of an HTMLTimingObject as basis for Web-based linear composition. A draft specification for the HTMLTimingObject is being developed by the W3C Multi-device Timing Community Group at http://webtiming.github.io/timingobject/.

The HTMLTimingObject is a very simple object, essentially an advanced stop-watch. If started, its value changes predictably in time, until at some point later, it is paused, or perhaps reset. The HTMLTimingObject may be queried for its value at any time. In terms of implementation, the HTMLTimingObject is a fairly trivial wrapping around the system clock. This means that it shares properties of the system clock in terms of resolution and predictability. This makes

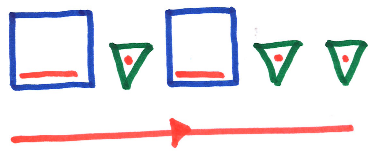

Figure 3: Media elements (blue rectangles) and track elements (green triangles) directed by a single HTMLTimingObject (red line).

the HTMLTimingObject a sound basis for precise linear synchronization.

The HTMLTimingObject is more expressive than a traditional stop-watch. It supports any velocity or acceleration, and may jump to any position on the timeline. In fact, the HTMLTimingObject simply implements linear motion along a unidimensional axis. An elegant implementation is provided by the concept of Media State Vectors (MSV) [2]. At any point in time, position, velocity or acceleration may be requested to change. Querying the HTMLTimingObject reveals not only its current value (position) but also its velocity and acceleration at that moment. This detailed information is again helpful in precise synchronization, and the expressiveness of the underlying mathematical model implies that a wide variety of control primitives may be supported.

We are not the first to define timing controls for linear media. Similar constructs have been explored in both academia and industry from the 70's and onwards. Indeed, any framework for linear media would maintain similar constructs internally. Instead, the novelty lies in representing timing as an explicit resource on the Web, independent of framework, thereby creating a basis for interoperability as well as multi-device timing. The latter is the subject of the next section.

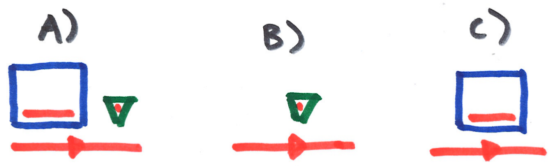

Figure 4: Media elements (blue rectangles) and track elements (green triangles) distributed or duplicated across devices, each device with its own HTMLTimingObject (red line).

Figure 3 illustrates linear composition, with the HTMLTimingObject (red line) replacing the HTMLMediaController as director. The illustration also shows how HTMLMediaElements and HTMLTrackElements equally interface directly with the timing object. Each of these linear components will monitor the timing object, and implement appropriate reactions whenever it pauses, resumes, jumps or speeds up. Linear components may also enforce control over the shared timing object, by requesting it to pause, resume etc. Such requests trigger effects on all components in unison. Crucially, all linear components may remain mutually independent, and even agnostic of each others existence. This is possible as they do not communicate directly, but only indirectly through a shared timing object. This independency between linear components is crucial for flexible and dynamic linear composition.

Multi-device Linear Composition for Web

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

Multi-device Regatta
Figure 4 may illustrate how a regatta could be presented in a multi-device scenario. 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, images and comments, while at the same time serving as an input-device for 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.

Figure 4 illustrates how two media elements (blue rectangles) and two track elements (green triangles) may be split across three devices. Note also that this multi-device scenario, particularly B), demonstrates why the current dependency between track elements and media elements is not appropriate. In this illustration, track elements are promoted as standalone programming constructs depending directly on the HTMLTimingObject, just like media elements.

In order to support multi-device linear composition, the challenge is to ensure that HTMLTimingObjects on multiple devices are kept in synchrony. In the single-device scenario, all linear components were using a single HTMLTimingObject as shared media control. We propose to extend this notion to the multi-device scenario.

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

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

Figure 5 illustrates a single, online timing object (red line), shared between distributed media elements (blue rectangles) and track elements (green triangles). The HTMLTimingObjects on each device (red lines) serve as local representation for the shared, online timing object. As the HTMLTimingObject encapsulates synchronization with online

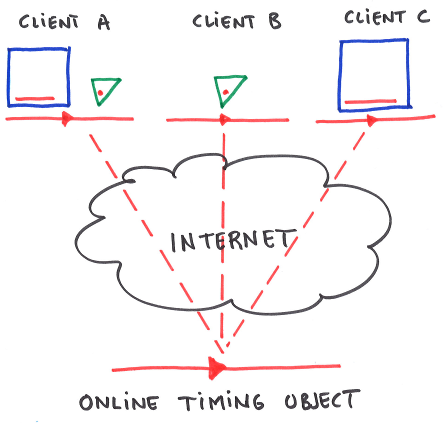

Figure 5: HTMLTimingObjects (red lines) mediating access to an online timing object (red line). HTMLTimingObjects on different devices connect and synchronize individually.

timing objects, media elements and track elements may readily support linear composition in multi-device as well as single-device media. In principle, distributed synchronization would only require the programmer to specify a valid URL for the source attribute of the HTMLTimingObject.

Shared Motion

A solution is already available for the implementation of synchronization with online timing objects. Shared Motion is a generic mechanism for distributed timing on the Web. Similar to the HTMLTimingObject, it is based on Media State Vectors (MSV) [2], a representation of deterministic, unidimensional motion. The synchronization protocol of Shared Motion is based on ad-hoc, application-level clock synchronization, and allows Web clients to maintain sub-framerate precision [2, 3] across the Internet. Shared Motion is also extremely scalable, allowing a vast number of clients to participate in multi-device linear media presentations. Shared Motion does not depend on NTP [8] synchronization, and works independent of network carrier, OS or browser type (without requiring plugins). Though Shared Motion is designed for the Web it is not restricted to the Web. As such, Shared Motion is an enabling mechanism for distributed linear composition, applicable to all connected clients, from web-browsers on laptop computers to native applications on embedded devices.

Evaluation

In order to evaluate HTMLTimingObjects and Shared Motion as basis for linear composition, we have focused on integration with HTMLMediaElement and HTMLTrackElements as a first step. A JavaScript wrapper is provided for multi-device synchronization of HTML5 Audio/Video, demonstrating synchronization errors < 10ms [3] across the Internet. A JavaScript re-implementation of the HTMLTrackElement demonstrates distributed presentation of timed data with millisecond precision (relative to HTMLTimingObject). The precision, reliability and simplicity of the proposed programming concepts further demonstrates the feasibility and value of this approach. Demonstrations have been made publicly available by Motion Corporation and the W3C Multi-device Timing Community Group.

In this work we have also identified a series of weaknesses with respect to HTML support for timed operation. We aim to address these weaknesses through the W3C Multi-device Timing Community Group, thereby improving the Web as a platform for timed media.

The HTMLMediaElement is currently not optimized for external control. In particular, media elements should compensate for internal delays (buffering etc.) by adjusting their internal playback offset according to an external HTMLTimingObject. In addition, to maintain synchrony, it is necessary for media elements to compensate for playback drift relative to the HTMLTimingObject. Media elements also behave differently with different browsers, different media types and on different architectures. It is hard to maintain a library masking all these differences. Even worse, properties affecting synchronization are sensitive to changes across browsers updates. So, ideally these problems should be addressed by media elements internally, thereby implementing native support for timed operation. The multi-device timing group will work on a test suite for HTMLMediaElements, exposing such timing issues.

The HTMLTrackElement too would benefit from improvements with respect to precision. Our initial tests indicate that cue events are emitted rather coarsely. In Chrome version 39 for example, events appear to fire about 150-250 milliseconds too late according to the video *currentTime*. Our improved version of the track element reliably delivers event upcalls at the correct millisecond. Furthermore, the current track element provides events only during playback, not as a response to seekTo. Our implementation guarantees enter and exit events to always be consistent with any kind of media control supported by Shared Motion, including acceleration. Consistency of enter and exit events additionally extends to dynamic cue changes. For instance, a timed subtitle may safely be added, removed, or prolonged in time- span, during playback, without introducing added complexity for the programmer.

If properly integrated with the HTMLTimingObject, media elements and track elements would be readily available for precise and flexible linear composition in the single device scenario. Additionally, integration of SharedMotion with the HTMLTimingObject would immediately enable support for multi-device linear composition.

Summary

This paper explains the importance of linear composition for Web-based multi-device media, and identifies precise, expressive multi-device timing control as the key technical enabler. The proposed solution involves the introduction of a HTMLTimingObject and integration with Shared Motion for multi-device timing. The approach is verified by integration of HTMLMediaElements and HTMLTrackElements with SharedMotion. The precision, reliability and simplicity of the proposed programming concepts support our claim that this approach unlocks a highly powerful programming model supporting multi-device linear composition on the Web.

References

[1] Adobe. 2015. Adobe Flash runtimes. https://www.adobe.com/products/flashruntimes.html. (2015).

[2] 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. DOI: http://dx.doi.org/10.1145/2457413.2457427

[3] Njål T. Borch and Ingar M. Arntzen. 2014. *Distributed Synchronization of HTML5 Media*. Technical Report 15. Northern Research Institute. http://norut.no/nb/node/3041

[4] Prezi Inc. 2015. Prezi for Presentation. http://prezi.com/. (2015).

[5] Jack Jansen and Dick C.A. Bulterman. 2008. Enabling Adaptive Time-based Web Applications with SMIL State. In *Proceedings of the Eighth ACM Symposium on Document Engineering (DocEng '08)*. ACM, New York, NY, USA, 18–27. DOI: http://dx.doi.org/10.1145/1410140.1410146

[6] Jack Jansen, Pablo Cesar, Rodrigo Laiola Guimaraes, and Dick C.A. Bulterman. 2012. Just-in-time Personalized Video Presentations. In *Proceedings of the 2012 ACM Symposium on Document Engineering (DocEng '12)*. ACM, New York, NY, USA, 59–68. DOI: http://dx.doi.org/10.1145/2361354.2361368

[7] Microsoft. 2015. Microsoft Silverlight. http://www.microsoft.com/silverlight/. (2015).

[8] David. L. Mills. 1991. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications* 39, 10 (Oct 1991), 1482–1493. DOI: http://dx.doi.org/10.1109/26.103043

[9] Mozilla. 2015. Popcorn.js The HTML5 Media Framework. http://popcornjs.org/. (2015).

[10] Patrick. Schmitz. 2002. Multimedia meets computer graphics in SMIL2.0: a time model for the web. In *Proceedings of the 11th international conference on World Wide Web (WWW '02)*. ACM, New York, NY, USA, 45–53. DOI: http://dx.doi.org/10.1145/511446.511453

[11] W3C. 2015a. HTML5 Media Controller. http://dev.w3.org/html5/spec-preview/Overview.html#synchronising-multiple-media-elements. (2015).

[12] W3C. 2015b. HTML5 Media Elements. http://dev.w3.org/html5/spec-preview/media-elements.html. (2015).

[13] W3C. 2015c. HTML5 Track Element. http://dev.w3.org/html5/spec-preview/the-track-element.html. (2015).

[14] W3C. 2015d. Web Animations. http://www.w3.org/TR/web-animations/. (2015).