Harbinger Group

Building an HTML5 Video Player with Custom Controls

A Harbinger Group White Paper

Harbinger Group

Harbinger Group

Your partner in technology innovation

About Harbinger Group: Harbinger Systems (www.harbingergroup.com) is a leading provider of software engineering services to some of the world's best product companies.

This white paper is part of Harbinger's Web Application practice. Harbinger has deep expertise in building consumer internet applications using various cutting-edge technologies in web and desktop. We bring years of experience in developing web and connected desktop applications that utilize the latest and most stable technologies in software development.

Visit our website to download or request our white papers on leading edge technologies and trends.

Contents

Introduction

The presence of video playback option on a web page is a common case and this can be achieved through plugins like Flash Player, Quicktime and SliverLight, etc. However, HTML5 Video Player avoids the use of any external plugin for video playback and opens the door to create multiple interactions between video and other elements within the page that has not been possible earlier. HTML5 is the base for web applications having multiple capabilities which include - Cascading Style Sheets and JavaScript. It is slowly but surely taking over online video space and the market data from StatCounter, a web analytics firm indicates that all the popular browsers as of today widely support HTML5 video specification.

HTML5 video player is technically a component built into HTML5 browsers. As of now, the key concern was the split support for video codecs, but now all modern browsers support the H.264 standard. However, the look and feel of the video controls is not consistent across all the browsers. The snapshot below certifies this point since we can see the same webpage in IE, Chrome and Firefox having same options but a different UI.

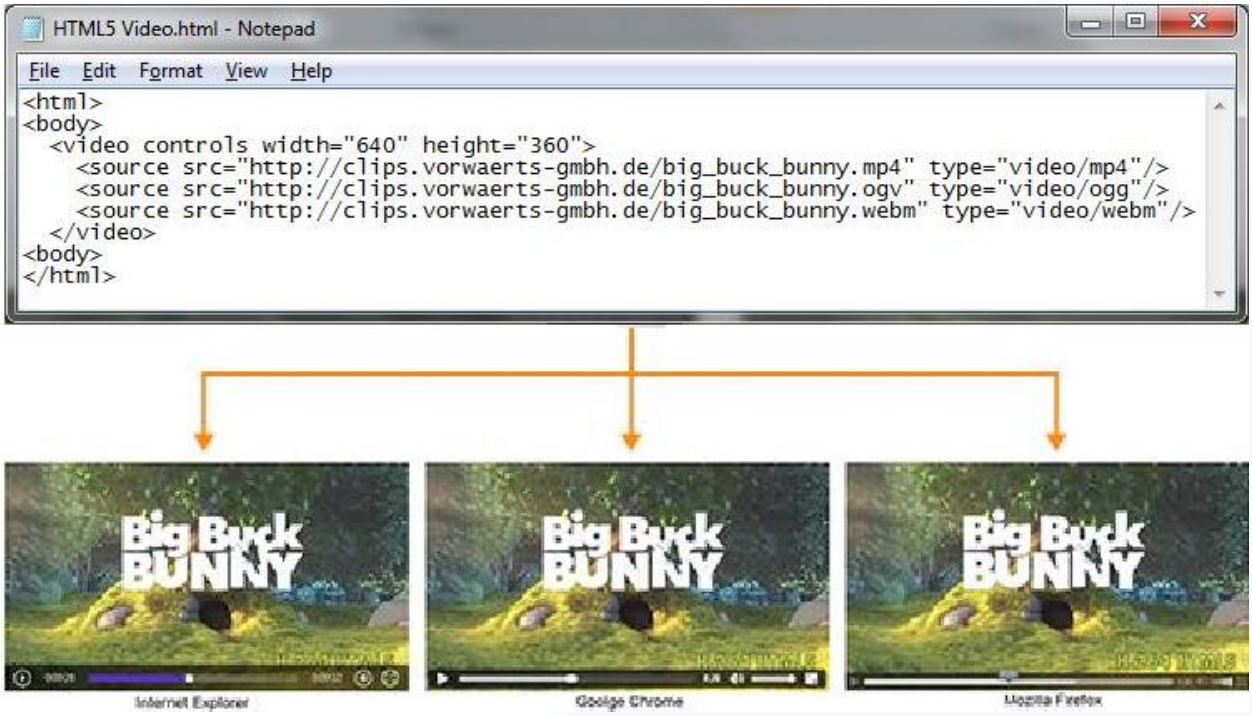

Fig 1: Inconsistency of video controls in HTML5 across browsers

This consistency can be achieved through an HTML5 player with JavaScript library which will control the look and feel of the video controls across all the browsers and provide access to all the video APIs as well.

How it works:

The HTML5 video tag contains two key attributes, 'src' and 'controls'. The 'src' attribute holds the URL which points to the video which is hosted on the web. Therefore, the control features decides the visibility/appearance of the on-board/default video player controls. In this, the source of the video tag which is present holds the URL - http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4'

The video tag supports three video formats which include - MP4, OGG and WebM. Out of these tags, MP4 is the most widely supported format by all modern HTML5 browsers.

- MP4: MPEG 4 files with H264 video codec and AAC audio codec
- Ogg: Ogg files with Theora video codec and Vorbis audio codec
- WebM: WebM files with VP8 video codec and Vorbis audio codec

In order to support different formats in the video tag, we just need to remove the 'src' attribute of the video tag and replace it with the 'type' attribute of the individual source tags.

The 'controls' attribute handles the visibility of the default video controls of the browser. If this attribute is omitted, then default video controls of the browser will not be visible.

Apart from the above key attributes, HTML5 video tag supports various other attributes and each feature offers different capabilities. For example, the 'poster' attribute allows specifying an image to be shown until the user plays the video. The 'autoplay' attribute allows playing of video as soon as it is ready without any user action.

The following table lists down which browser version supports the different attributes of HTML5 video tag:

Browser	Attributes	Poster	Controls	Preload	Autoplay	Loop	Muted
Desktop	Firefox Version	20+	20+	20+	25+	25+	25+
	Chrome Version	30+	30+	20+	30+	30+	30+
	Internet Explorer Version	10+	9+	10+	9+	9+	9+
Mac	Safari Version	3+	3+	5+	3+	3+	3+
Windows Phone	Internet Explorer Version	7.5+	7.5+	7.5+	8.1	8.1	8.1
iOS	Safari Version	3+	3+	3+	-	-	-
Android	Stock Browser Version	2.3+	4.0+	4.0+	-	-	-
	Chrome Version	30+	30+	30+	-	-	-
	Firefox Version	20+	32+	20+	32+	32+	32+

Poster and Preload attributes behavior in Internet Explorer 9:

- The poster shows only when preload is set to none
- Always pre-fetches the entire video, regardless of the preload setting

HTML5 Video Player

In this section, we will explain how HTML5 player with custom controls can be created with the following set of features:

- Play and Pause Control
- Video Seek bar
- Volume Control
- Fullscreen Control

Further, we will describe how a HTML5 player can be extended to play videos from YouTube, Vimeo and Kaltura platforms.

The custom HTML5 player consists of three layers which include:

- **Player Layout** – This layer consists of a HTML structure for the Player including necessary player controls. It offers overall control on the look and feel of the Player with the help of style sheet and consumes JavaScript APIs exposed by the Player Core

- **Player Core** – This is the core layer of the Player consisting of JavaScript APIs. This layer exposes JavaScript APIs for different video elements like play, pause, volume, total video duration, current video time, etc. This layer internally consumes different wrapper APIs based on the type of video

- **Video Delivery Platforms** – The video source file hosted on CDN platforms are streamed directly from the respective CDN platform by this layer

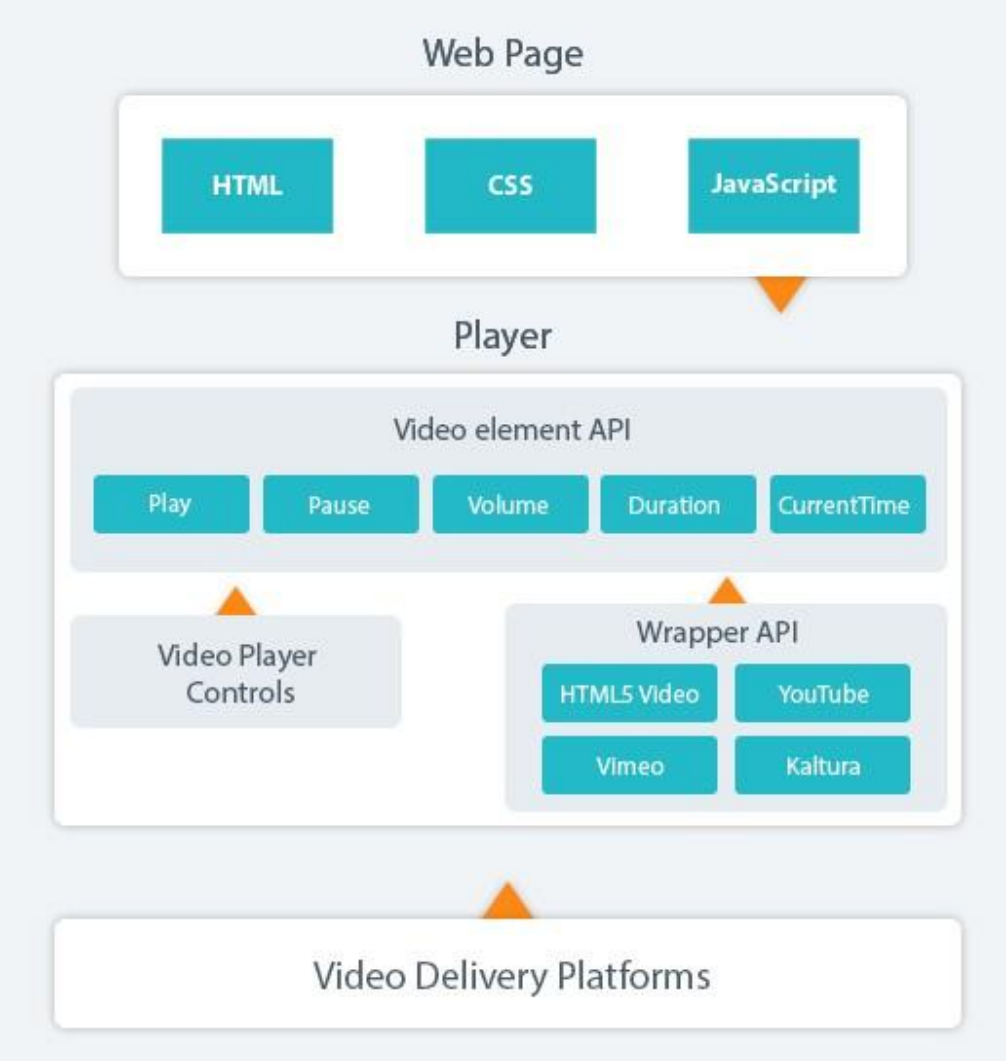

Fig 2: Architecture of HTML5 video player

Having seen the high level architecture, we will now elaborate on how custom control functionality can be implemented within the video player.

Play and Pause Control: A common method can be created to toggle the video. This method can internally detect the current state of the video based on which it either plays or pauses the video.

Video Seek bar: The JQuery UI range slider can be used for implementing a video seek bar. For the the range slider, we need to provide the total duration of the video as the maximum parameter.

Volume Control: A common method can be created to set and retrieve the volume. If the volume parameter is passed then it sets the volume or else the method returns to the current volume. The volume parameter scale range can be decided as per the requirements, usually, its generic range is from 0 to 1.

Fullscreen Control: The HTML5 Fullscreen API can be used to implement this feature. A common method is used to toggle the video fullscreen, this method internally detects the current state of the video fullscreen based on which it either enters the video into fullscreen mode or exits the video from fullscreen mode.

These custom controls ensure the look and feel of the player control to remain consistent across all the browsers, as seen in the screen snapshot below. (Reference: link http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4)

Fig 3: Consistent controls in HTML5 video player

Things to remember

Following are some of the few generic guidelines which ensure player size and custom controls remains intact in all scenarios:

- The z-index for html elements of the custom player controls has to be set to 2147483647. This will keep the custom player controlsvisible when the video is in fullscreen mode
- If the video is embedded using HTML5 video tag then the following style sheet needs to be added to hide the browser native video controls in the fullscreenmode

```
video::-webkit-media-controls {
display:none !important;
}
```

The viewport meta tag needs to be added in the player HTML to keep the player in its original size in the mobile devices

<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0">

Packaged HTML5 video players

There are many app developers in the industry who readily offer HTML5 video players. The following Table lists down some of the well-known HTML5 video players and compares them on different Parameters: (reference: http://praegnanz.de/html5video/)

Attributes / Players	License	JavaScript Library	Flash Fallback	iOS Support	Fullscreen	Keyboard Interface	Subtitles Support
Video JS	GPL v3	Stand alone	✓	✓	✓	-	✓
JW Player	Custom		✓	✓	✓	-	✓
Flow Player	Free version GPL v3	JQuery	✓	✓	✓	✓	✓
Projekktor	GPL v3	JQuery	✓	✓	✓	✓	✓
JMediaelement	GPL v2 and MIT	JQuery	✓	-	✓	✓	✓
Mediaelement.js	GPL v2 and MIT	JQuery	✓	✓	✓	-	✓
SublimeVideo	Service (Free Option)	Stand alone	✓	✓	✓	✓	✓
LeanBack	GPL v3	Stand alone	-	✓	✓	✓	✓

Integration with other video platforms

In this section, we will explain how to extend the capabilities of a custom video player to enable it for playing the videos hosted across different video delivery platforms. There are many video delivery platforms available but we will cover the most popular ones in this whitepaper.

YouTube

YouTube provides IFrame Player API using which YouTube videos can be embedded. This API provides seamless interface with YouTube player and also offers the ability to control YouTube video events, properties and methods.

The HTML5 video player needs to implement 'onYouTubeIframeAPIReady' JavaScript function. The Iframe API invokes this function once the JavaScript for player API is loaded. Internally, this method can initialize the YouTube player object using YouTube video ID which would load the video.

The API offers 'onReady' and 'onStateChange' events. The 'onReady' event gets triggered when the video player is ready to play the video and 'onStateChange' event gets triggered when the player states the changes, which indicate that the player is playing, paused, finished and so forth. The detailed information about IFrame API is available on Google developer portal. https://developers.google.com/youtube/iframe_api_reference

Vimeo

Vimeo provides JavaScript Player API which gives the ability to control the embedded Vimeo player. The API offers a window.postMessage() mechanism to interact with the player and the same can be achieved by sending a serialized JSON object with postMessage(). The format of the JSON object is as follows:

```
{
    "method": "methodName",
    "value": "value"
}
```

If a method does not accept the value then we need to omit the value key. As the postMessage() mechanism is complex, Vimeo offers a JavaScript mini library called Froogaloop that acts as a wrapper over the postMessage() and makes life easier for the developer. The library reference can be added as follows: *<script src="https://f.vimeocdn.com/js/froogaloop2.min.js"></script>*

The API also offers 'loadProgress' and 'playProgress' events. In this functioning, 'loadProgress' event gets triggered when the video is loading; it shows the current loaded percentage and the duration of the video. The 'playProgress' event gets activated when the video is playing; it shows the seconds, percentage exhausted and the total duration of the video. The detailed information about JavaScript API is available on Vimeo developer portal. https://developer.vimeo.com/player/js-api

Please note, the Vimeo API does not provide the mechanism to override or hide the default Vimeo video player controls. The default Vimeo video player controls can be controlled by the owner of the video from the Vimeo portal. This feature is available for the users of Vimeo Plus or Pro account.

Kaltura

Kaltura provides a kWidget API using which a standalone video URL can be retrieved for the videos hosted in Kaltura. The video URL then can be used as a source in the standard HTML5 video tag to embed the video. The kWidget API returns multiple video URLs having different sizes and bitrates.

The kWidget API is available after we include it in the Kaltura player API library. JavaScript way of including Kaltura Player API library is as follows:

```
<!-- Substitute {partner_id} for your Kaltura partner id, {uiconf_id} for uiconf player id -->
<script
src="http://cdnapi.kaltura.com/p/{partner_id}/sp/{partnerId}00/embedIframeJs/uiconf_id/{uiconf_id}/partner_id/{partnerId}"></script>
```

- The 'getSources' method of kWidget API accepts partnerId and entryId as input parameters and fires a callback method which can capture the video sources information. The 'mediaReady' event of the API gets triggered when the player is ready to play the video. The detailed information about the API is available on Kaltura portal http://player.kaltura.com/docs/index.php?path=api
- Similarly, the other video delivery platforms can be integrated seamlessly with the custom video player, just by plugging the necessary APIs offered by the video platforms.

Mobile Challenges

Today, most of the mobile browsers are developed having HTML5 compatibility. HTML5 has upgraded and advanced in the sector of Animation and Video but there is a huge scope of improvement for mobile based browsers. However, recently developed mobile browsers do have adequate support for HTML5 video tag specifications. Also, the screen size and network bandwidth of a mobile device is small and limited when compared with desktop computers. These two factors highlight the limitations on some of the attributes of the HTML5 video tag as listed below:

- **Autoplay** or **preload** is not supported on mobile devices. This means on mobile devices video will not start playing or buffering till the time the user performs some action.
- **Volume** property of video is read-only on mobile devices. This means on mobile devices the user will not be able to control video volume using custom controls. The video volume will be controlled by the user using device volume control.
- **HTML5 Fullscreen** – iOS Safari browser and Android Stock browser does not support HTML5 Fullscreen APIs
- **iPhone** does not play video inline in the webpage, it plays the video in fullscreen mode using native video controls. Therefore, the custom video player controls will not be visible on the iPhone screen. Apple provides webkit-playsinline attribute for video tag to play video inline in webpage, however, as observed by us, this attribute works only when the webpage is bookmarked to iPhone home screen. The native video player application with custom controls can be built for the iPhone with the help of MPMoviePlayerController class reference.

As we move ahead with the continuous evolvement of HTML5 video tag specification we can experience that the player with HTML5 capabilities is emerging to be as a widely used standardized common for delivering exceptional videos across browsers and devices. The three layered architecture ensures new controls can be easily programmed in the custom HTML5 video player. It also provides scalability to include other available delivery platforms.

Contact us now, to know more about our services in the HTML5 video player space.

Glossary

- **AAC:** Advanced Audio Coding (AAC) is an audio coding standard for lossy digital audio compression
- **CDN:** A content delivery network or content distribution network (CDN) is a large distributed system of servers deployed in multiple data centers across the Internet
- **Froogaloop:** Small JavaScript utility framework to help with the complexities involved in dealing with using the JavaScript API through window.postMessage for controlling Vimeo's Moogaloop video player
- **GPL:** The GNU General Public License (GNU GPL or GPL) is the most widely used free software license, which guarantees end users (individuals, organizations, companies) the freedoms to run, study, share (copy), and modify the software
- **H.264:** Advanced Video Coding (MPEG-4 AVC) is a video coding format that is currently one of the most commonly used formats for the recording, compression, and distribution of video content
- **JSON:** JavaScript Object Notation is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.
- **kWidget:** HTML5 Kaltura JavaScript API library
- **MP4:** Digital multimedia format most commonly used to store video and audio, but can also be used to store other data such as subtitles and still images
- **MPMoviePlayerController:** A movie player manages the playback of a movie from a file or a network stream
- **OGG:** Free, open container format maintained by the Xiph.Org Foundation. The creators of the Ogg format state that it is unrestricted by software patents and is designed to provide for efficient streaming and manipulation of high quality digital multimedia.
- **VP8:** Video compression format owned by Google and created by On2 Technologies as a successor to VP7
- **WebM:** Video file format, primarily intended to offer a royalty-free alternative to use in the HTML5 video tag. The development of the format is sponsored by Google, the corresponding Software is distributed under a BSD license.

About Harbinger Group

Harbinger Systems is a global company providing software technology services for independent software vendors and enterprises, with a specialization in product engineering. Since 1990, Harbinger has developed a strong customer base worldwide. Harbinger's customers are software product companies, including hi-tech startups in Silicon Valley, to leading product companies in the US and large in-house IT organizations.

Harbinger Systems builds software solutions leveraging social, mobile, analytics, and cloud (SMAC) technologies for domains such as human capital management (HCM), healthcare, e-learning, and publishing. Harbinger Systems also uses emerging technologies like big data, OpenStack, and Internet of Things (IoT) to build products for tech startups.
By leveraging cutting-edge technologies, Harbinger Systems works with its customers as a partner in technology innovation.

Services

- Software Product Development
- Mobile App Development
- Internet of Things
- eLearning
- Advanced Testing
- Systems Software

Awards and Recognitions

To learn more about our awards Click Here

Visit us at: www.harbingergroup.com

Follow us on:

Software Technology blog | **Twitter** | **Facebook** | **LinkedIn** | Check our presentations on **SlideShare**