Google™ 11 IO

YouTube's iframe Player: The Future of Embedding

Jeffrey Posnick, Greg Schechter, and Jarek Wilkiewicz
May 11, 2011

Session Overview

- What is the iframe player?
- HTML5 playback in detail.
- Details of writing and exposing a JavaScript API for controlling the player.
- Comparing the iframe API to the AS3 player API.
- iframe player API example application.

Hashtags: #io2011 #YouTube
Feedback: http://goo.gl/fdY2L

Session Overview

- What is the iframe player?
- HTML5 playback in detail.
- Details of writing and exposing a JavaScript API for controlling the player.
- Comparing the iframe API to the AS3 player API.
- iframe player API example application.

“Let the embed, not the embedder figure it out!”

Video on the Web is getting complex

What is the iframe player?

- Problem: platform fragmentation
  - PC vs Mobile
  - Encoding: H.263, H.264, WebM/VP8, ...
  - RTSP/AS2/AS3/HTML5
- Solution: <**iframe**> player
  - Let the embed, not the embedder figure it out
  - Common API independent of video technology

```
<iframe class="youtube-player" type="text/html"
width="640" height="385" src="http://www.youtube.
com/embed/ID>
</iframe>
```

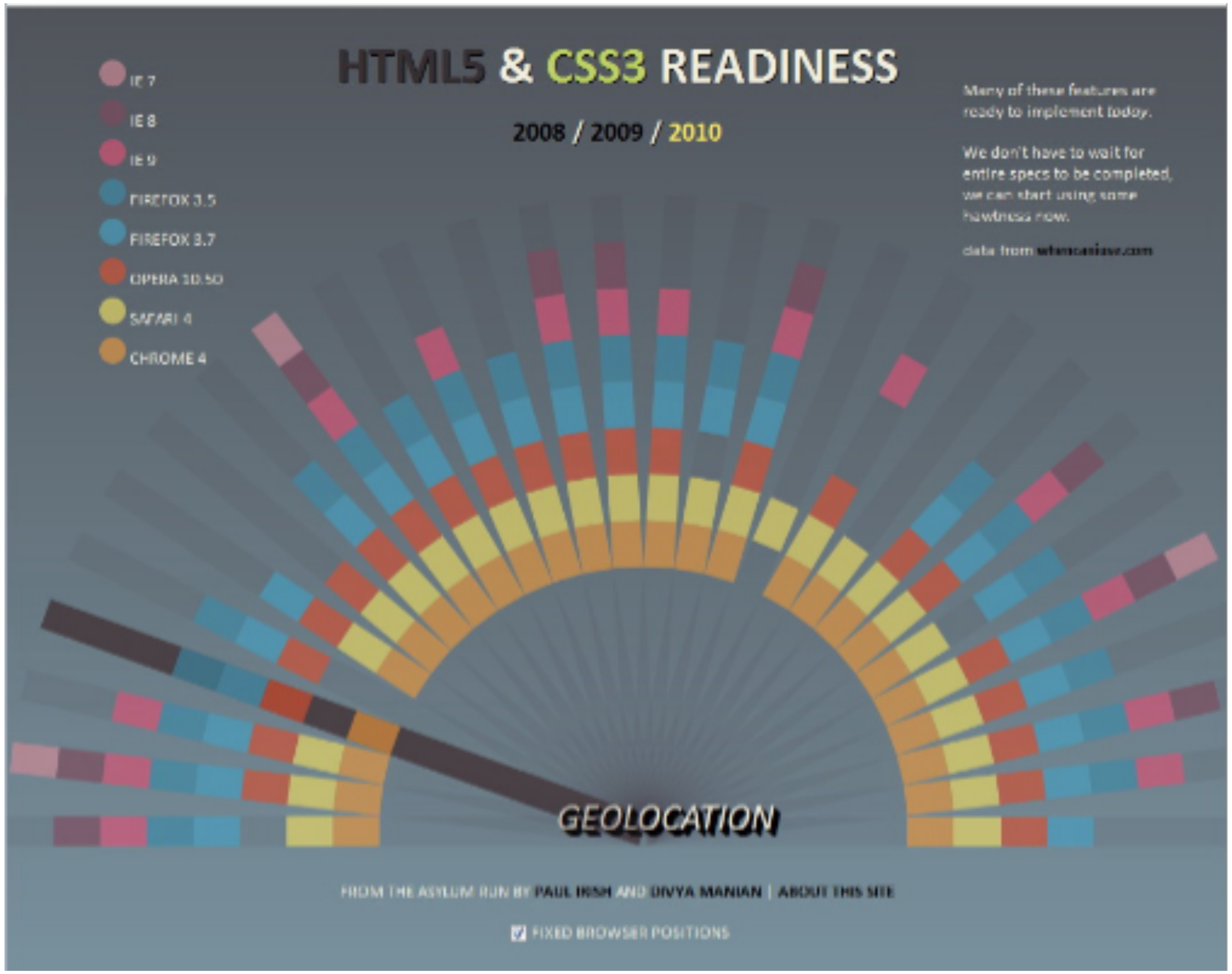
Session Overview

- What is the iframe player?
- HTML5 playback in detail.
- Details of writing and exposing a JavaScript API for controlling the player.
- Comparing the iframe API to the AS3 player API.
- iframe player API example application.

Why HTML5?
HTML5 vs. Flash

- Performance
- Accessibility
- Device-ability
- Features
- Security
- Embeds API

Why HTML5?
Features



http://www.flickr.com/photos/zipckr/4624150058/

Why HTML5?

Features

- Robust video streaming

Why HTML5?
Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video

Why HTML5?

Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video
- Content protection

Why HTML5?
Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video
- Content protection
  - RTMPE protocol / Flash Access

Why HTML5?

Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video
- Content protection
  - RTMPE protocol / Flash Access
- Fullscreen video

Why HTML5?
Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video
- Content protection
  - RTMPE protocol / Flash Access
- Fullscreen video
  - We need HD cat videos!
  - WebKit has a JavaScript API

Why HTML5?

WebKit Fullscreen API

```
var elem = document.getElementById("my-element");
elem.onwebkitfullscreenchange = function() {
console.log ("We went fullscreen!");
};
elem.webkitRequestFullScreen();
```

Why HTML5?
Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video
- Content protection
  - RTMPE protocol / Flash Access
- Fullscreen video
  - We need HD cat videos!
  - WebKit has a JavaScript API
- Camera & Microphone Access

Why HTML5?
Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video
- Content protection
  - RTMPE protocol / Flash Access
- Fullscreen video
  - We need HD cat videos!
  - WebKit has a JavaScript API
- Camera & Microphone Access
  - Need to record to YouTube

Why HTML5?
Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video
- Content protection
  - RTMPE protocol / Flash Access
- Fullscreen video
  - We need HD cat videos!
  - WebKit has a JavaScript API
- Camera & Microphone Access
  - Need to record to YouTube
- Formats

Why HTML5?
Features

- Robust video streaming
  - Fine control over buffering and dynamic quality control
  - Jump to any part of the video
- Content protection
  - RTMPE protocol / Flash Access
- Fullscreen video
  - We need HD cat videos!
  - WebKit has a JavaScript API
- Camera & Microphone Access
  - Need to record to YouTube
- Formats
  - Need to support both H.264 and WebM

Why HTML5?
<video> Expectations

- Open source technology
    - Browser / Player / Codec

Why HTML5?
<video> Expectations

- Open source technology
  - Browser / Player / Codec
- Lower latency
  - No plug-in instantiation

Why HTML5?

<video> Expectations

- Open source technology
  - Browser / Player / Codec
- Lower latency
  - No plug-in instantiation
- Better performance and fidelity

Why HTML5?
<video> Expectations

- Open source technology
  - Browser / Player / Codec
- Lower latency
  - No plug-in instantiation
- Better performance and fidelity
- Accessibility

Why HTML5?

<video> Expectations

OH, HI; I'M HERE FROM THE INTERNET.
WHAT ARE YOU DOING!?
GLUING CAPTIONS TO YOUR CATS.
RARR

http://imgs.xkcd.com/comics/in_ur_reality.png

Why HTML5?
<video> Expectations

- Open source technology
  - Browser / Player / Codec
- Lower latency
  - No plug-in instantiation
- Better performance and fidelity
- Accessibility
  - User agents can have special video handling

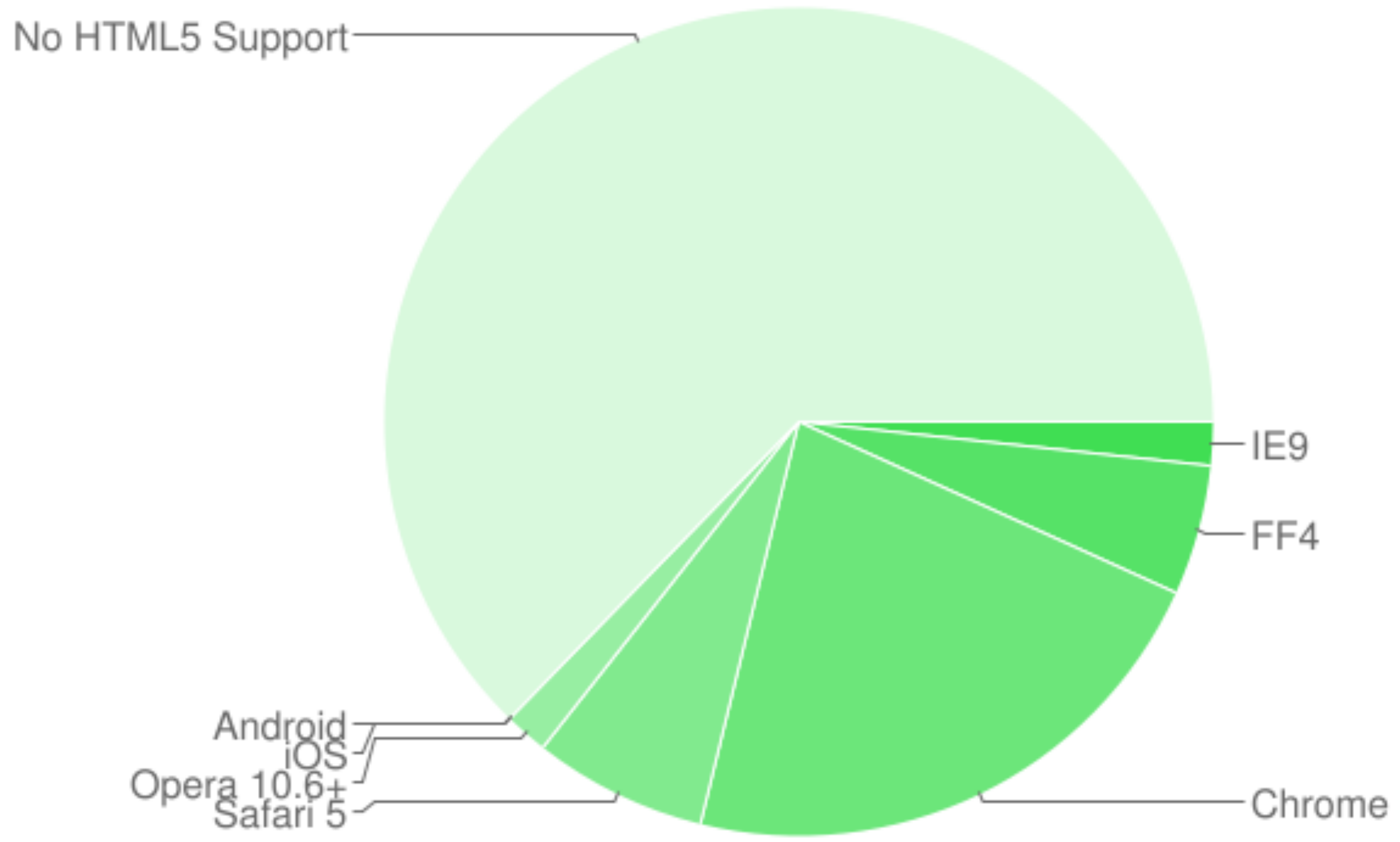
Why HTML5?

Device-ability

By cambodia4kidsorg
http://www.flickr.com/photos/cambodia4kidsorg/5228268296/

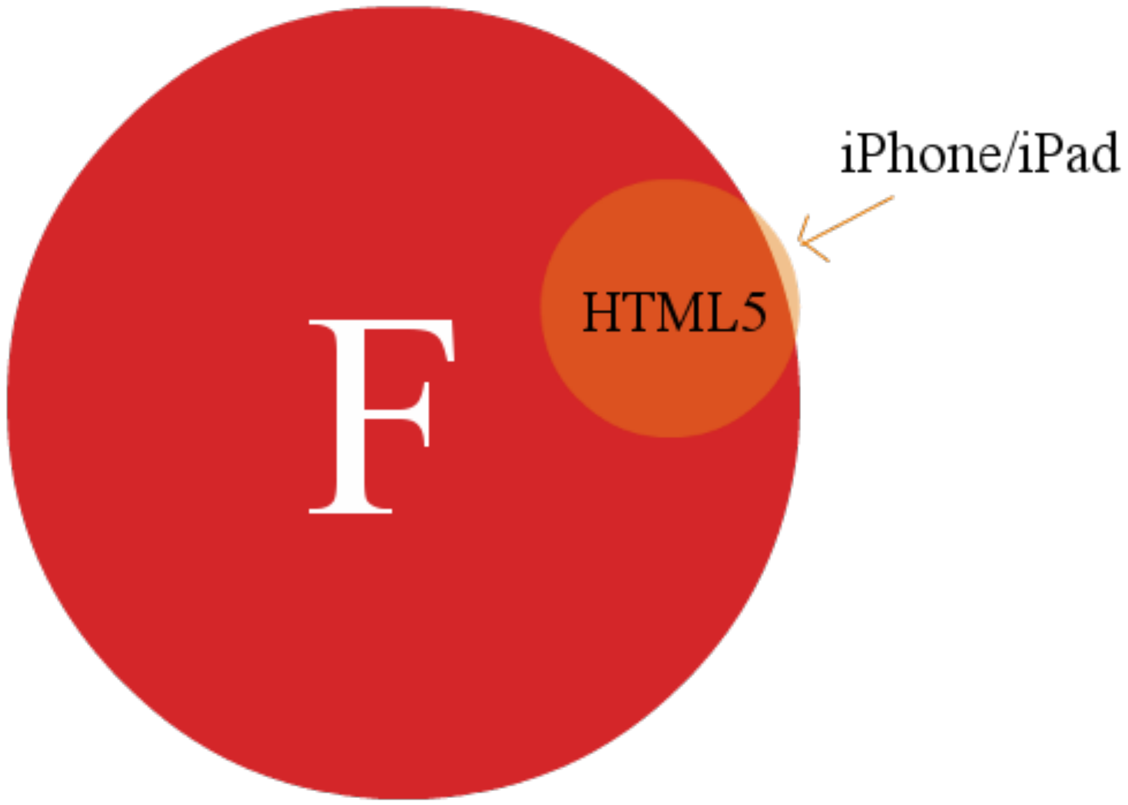
Why HTML5?

HTML5 Capable Browsers



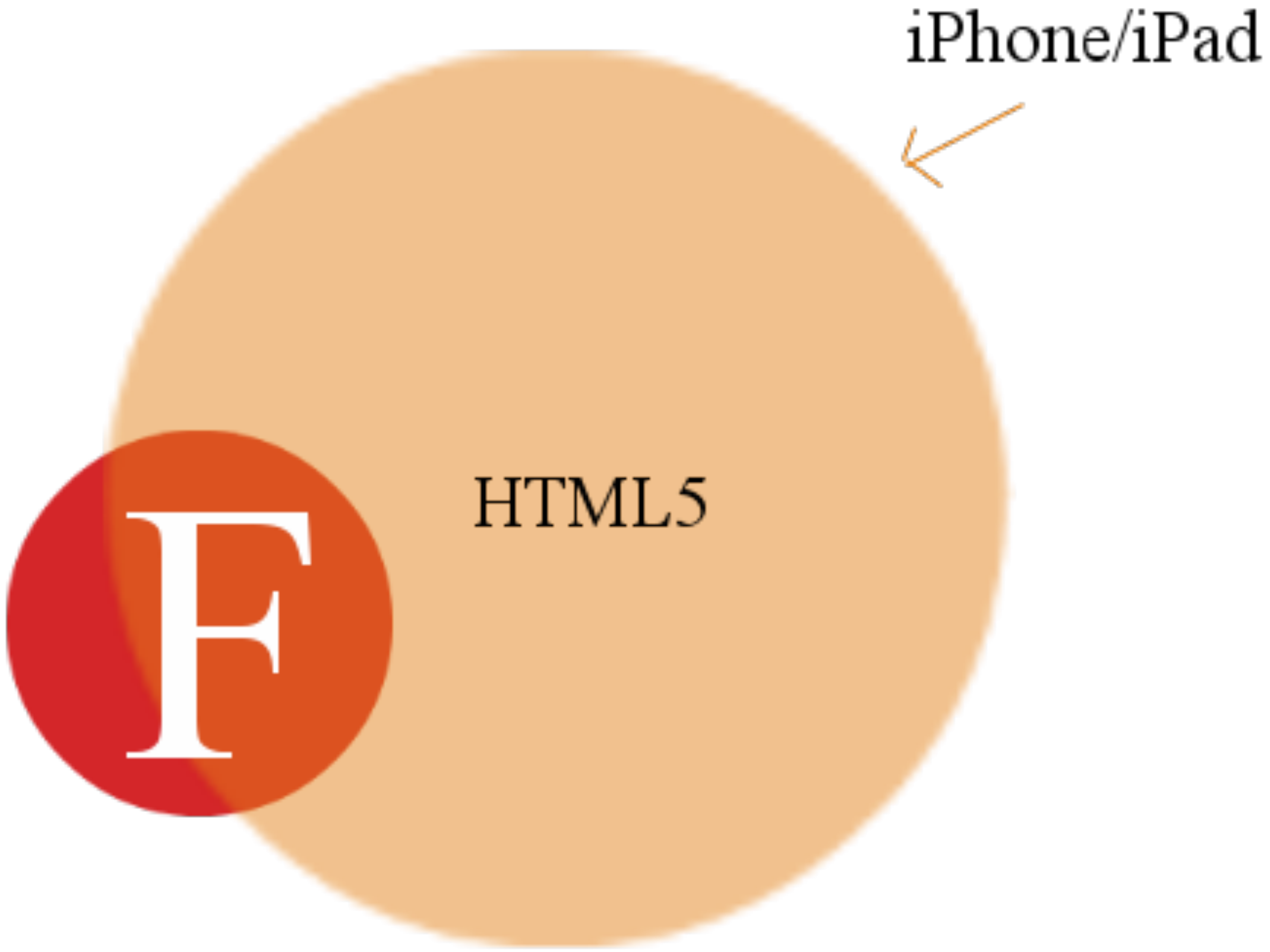
Why HTML5?

Flash Support vs. HTML5 Support



Why HTML5?

YouTube Data API Usage for Flash vs. HTML5 Devices



Why HTML5?

When HTML5?

When HTML5?

- Primary goal: Recover playbacks that would be lost without Flash.

When HTML5?

- Primary goal: Recover playbacks that would be lost without Flash.
- Our solution:

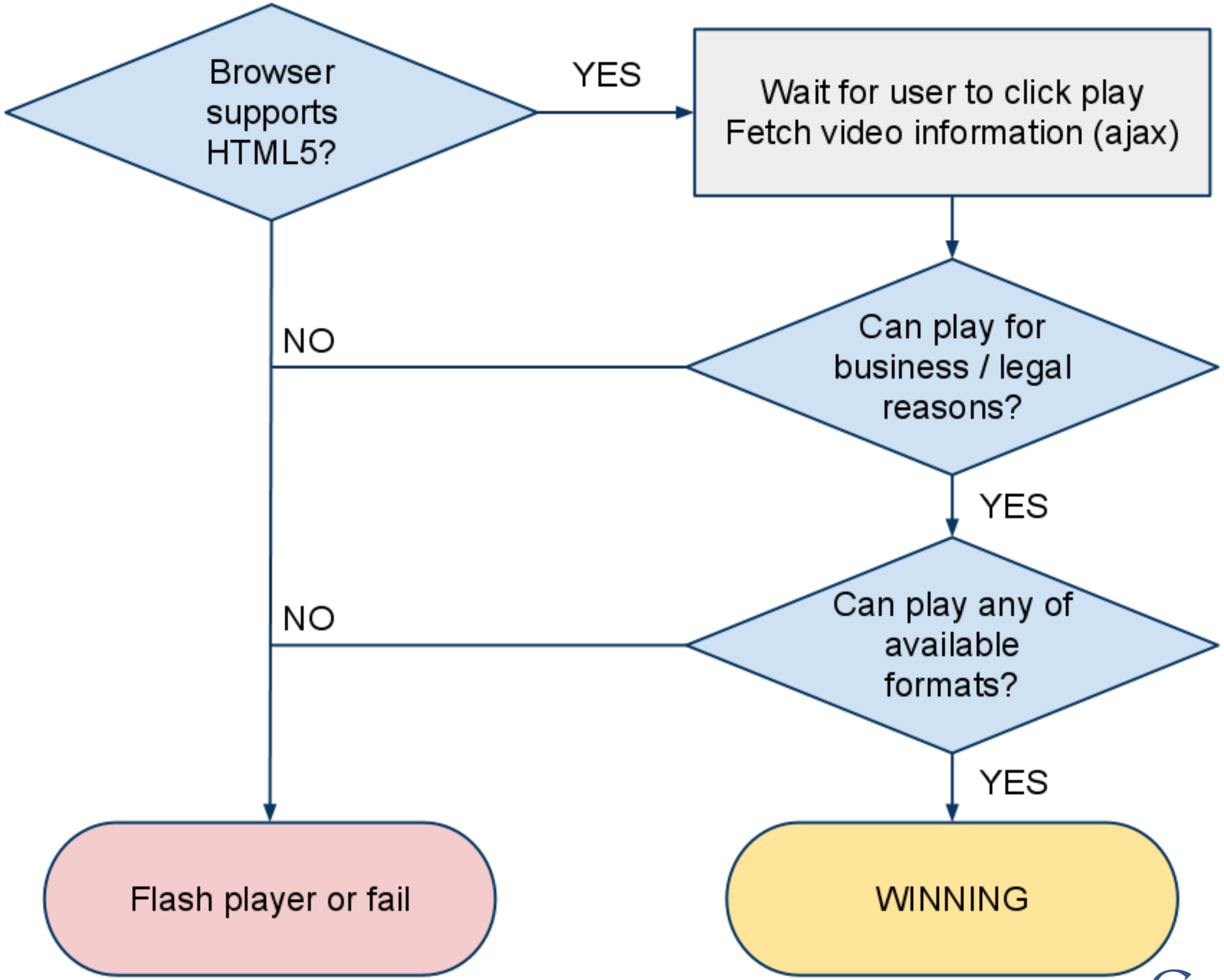
```
<iframe type="text/html"
  width="640"
  height="385"
  frameborder="0"
  src="http://www.youtube.com/embed/VIDEO_ID"
  allowfullscreen>
</iframe>
```

When HTML5?
<iframe> Embed

- Give the user HTML5 or Flash based on device and user preferences.
- Allows for better mobile support.
- Offers an "it just works" experience.

When HTML5?

When does the user get HTML5?


Browser supports HTML5?
YES
Wait for user to click play
Fetch video information (ajax)
NO
Can play for business / legal reasons?
YES
NO
Can play any of available formats?
YES
Flash player or fail
WINNING

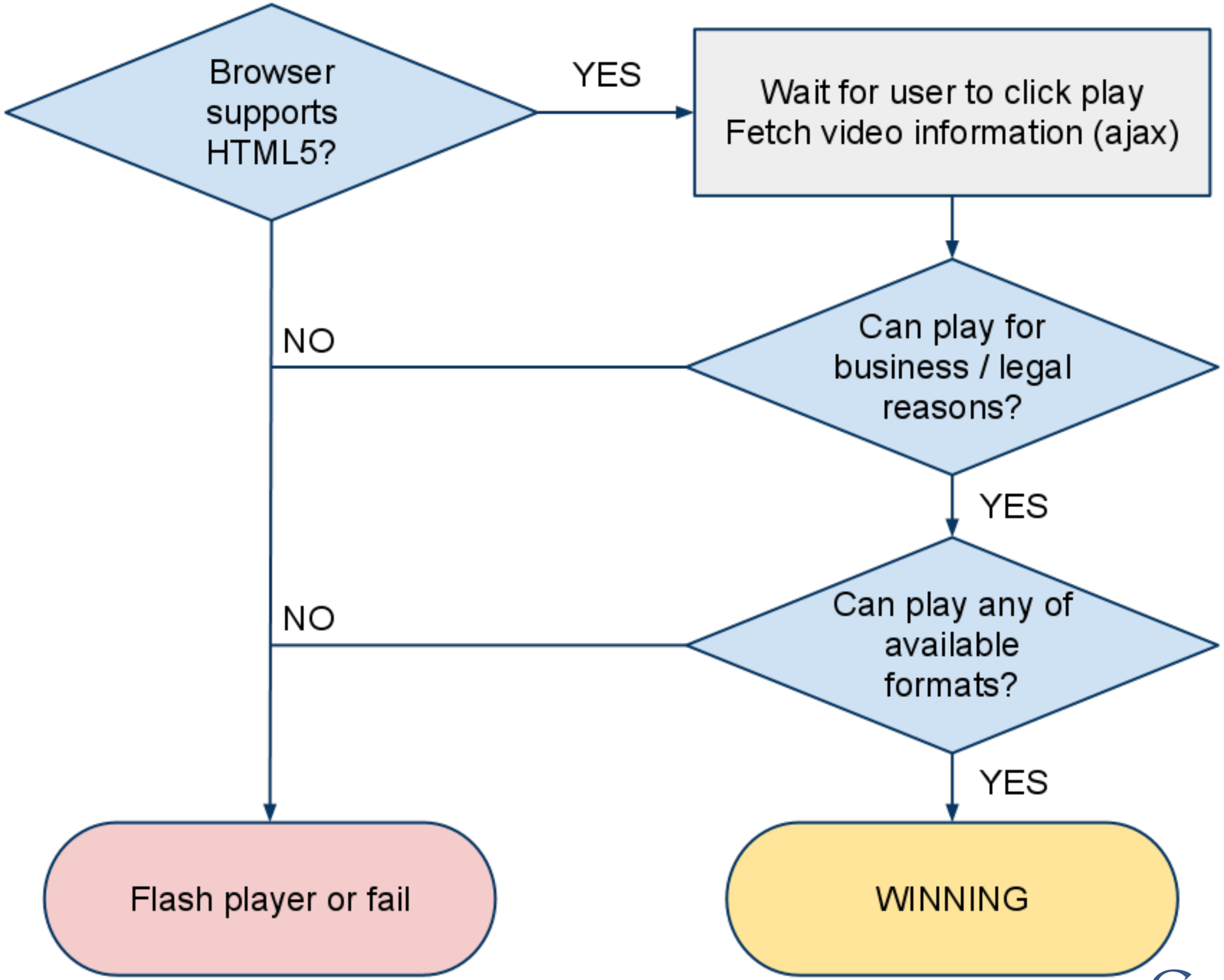
When HTML5?

Detecting HTML5

```
var videoElement = document.createElement('video');
if (videoElement && videoElement.canPlayType &&
   (videoElement.canPlayType('video/mp4; codecs="avc1.42001E, mp4a.40.2"') ||
    videoElement.canPlayType('video/webm; codecs="vp8.0, vorbis"'))) {
  // Sweet, we can use HTML5!
}
```

When HTML5?

When does the user get HTML5?

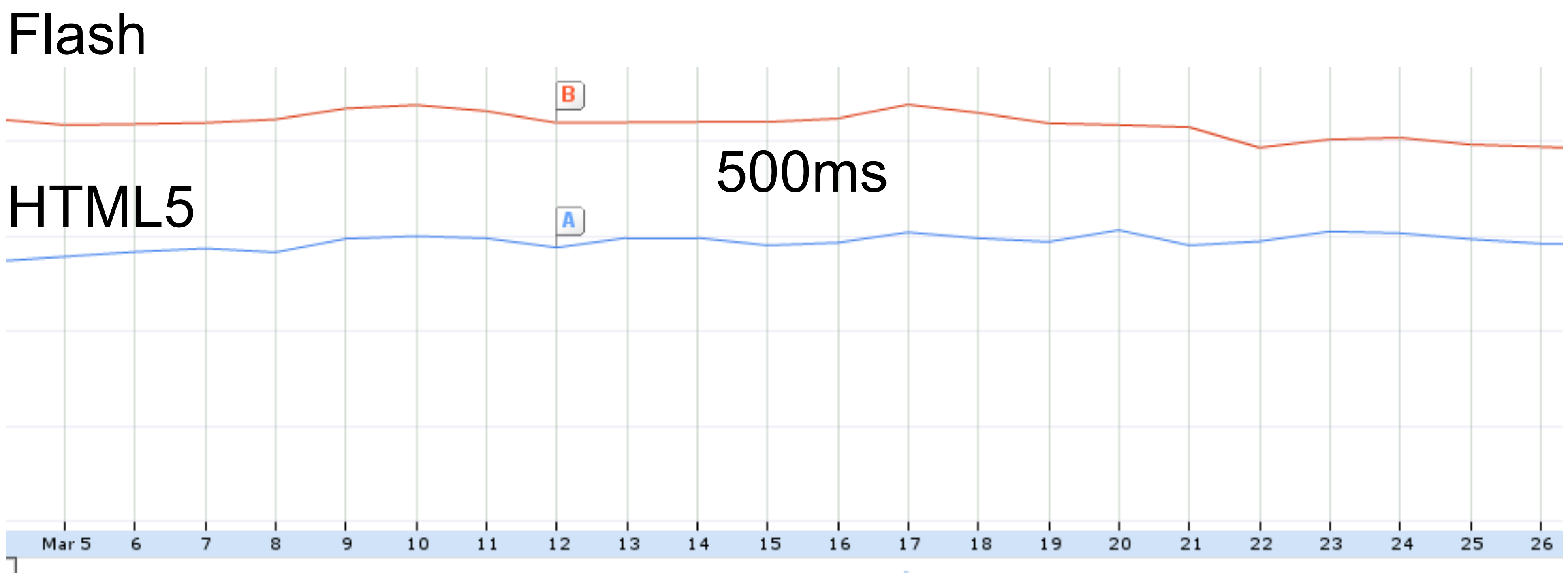

Browser supports HTML5?
YES
Wait for user to click play
Fetch video information (ajax)
NO
Can play for business / legal reasons?
YES
NO
Can play any of available formats?
YES
Flash player or fail
WINNING

Performance

By Two Hawk's Eye
http://www.flickr.com/photos/mycoolpics/92033686/

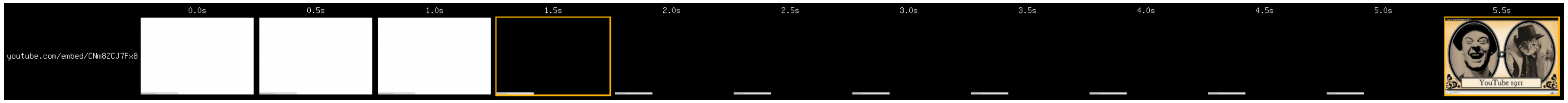
Performance

Player Start Time

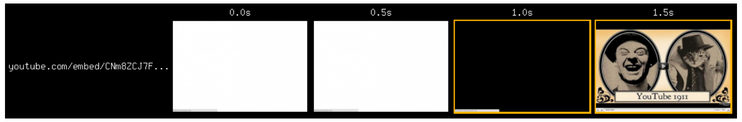


Performance
Time Until Thumbnail is Visible

Flash - 5.1s



HTML5 - 1.4s



*Collected data shows faster load times than this control environment, but the comparison is actuate.

Session Overview

- What is the iframe player?
- HTML5 playback in detail.
- Details of writing and exposing a JavaScript API for controlling the player.
- Comparing the iframe API to the AS3 player API.
- iframe player API example application.

The JavaScript API

Communication

The JavaScript API
Communication

- Poll the URL fragment?

    http://youtube.com/embed/video_id#**fragment**

The JavaScript API

Communication

- Poll the URL fragment?

http://youtube.com/embed/video_id#**fragment**

- Messages are one dimensional.
- Polling eats up CPU and is not instant.
- Both directions of communication use the same fragment.

The JavaScript API

Communication

- Better idea: PostMessage API.

```
someWindow.postMessage(message, targetOrigin);
```

The JavaScript API
Communication

- Better idea: PostMessage API.

```
someWindow.postMessage(message, targetOrigin);
```

- Uses JSON for native encoding and decoding of data.
- No polling.
- Native event listeners.
- Communications are sandboxed per-window.
- Calls are asynchronous.

Session Overview

- What is the iframe player?
- HTML5 playback in detail.
- Details of writing and exposing a JavaScript API for controlling the player.
- Comparing the iframe API to the AS3 player API.
- iframe player API example application.

Comparing the iframe API to the AS3 Player API

Three Ways to Control the Player

- Player Params
- Action Script API
- JavaScript API

Comparing the iframe API to the AS3 Player API

Player Params

- Example

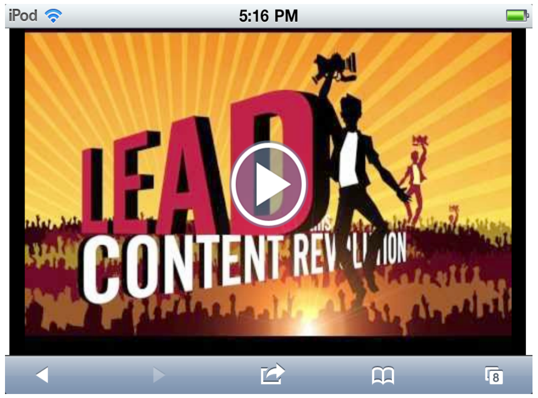
```
<iframe class="youtube-player" type="text/html"
width="640" height="385" src="http://www.youtube.com/embed/ID?
autoplay=1">
</iframe>
```

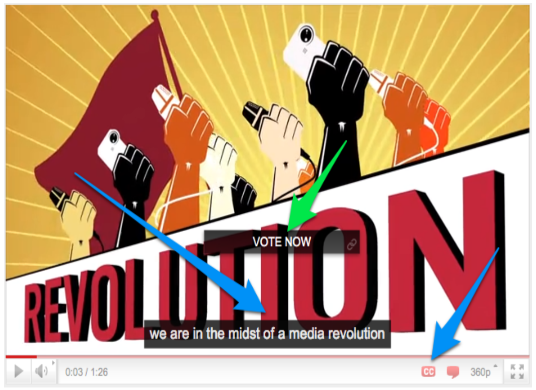
- AS3/Flash implementation of iframe : pass through

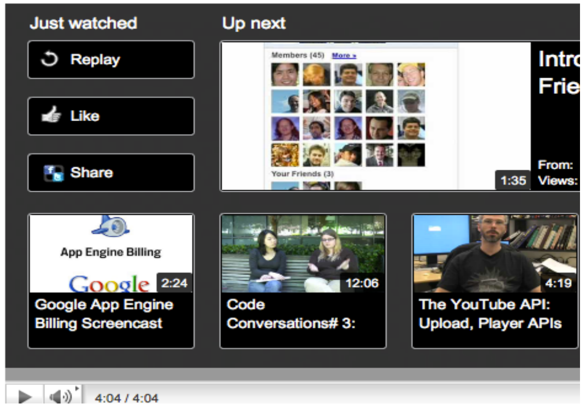
Comparing the iframe API to the AS3 Player API

Not (Fully) Supported by HTML5 Player Yet

- Autoplay - on iOS need to hit play (autoplay)
- Captions (no ASR, can't force cc_load_policy)
- Full screen (fs)
- Annotations (iv_load_policy)
- Related videos (rel)







Comparing the iframe API to the AS3 Player API

ActionScript API

- Not applicable!
- No ActionScript API for the iframe
- Use Flash Player API
- ... but this won't work on iOS :(
- Let's talk JavaScript

Comparing the iframe API to the AS3 Player API

JavaScript API

- Player Init
- Operations
- Event Handling

Comparing the iframe API to the AS3 Player API

Player Init

```
//Load player api asynchronously.
var tag = document.createElement(
         'script');
tag.src = "http://www.youtube.com/
          player_api";
var firstScriptTag = document.
          getElementsByTagName(
          'script')[0];
firstScriptTag.parentNode.
          insertBefore(tag,
          firstScriptTag);
var player;

function onYouTubePlayerAPIReady() {
  player = new YT.Player('player', {
     height: '390', width: '640',
     videoId: 'exmwSxv7XJI',
     playerVars: { 'autoplay': 1},
     events: {
      'onReady': onPlayerReady,
      'onStateChange':
            onPlayerStateChange
      }
  });
}
```

```
<script type="text/javascript" src="swfobject.js"
></script>

var params = { allowScriptAccess: "always" };
var atts = { id: "myytplayer" };

swfobject.embedSWF("http://www.youtube.
com/e/exmwSxv7XJI?
enablejsapi=1&        playerapiid=ytplayer&autoplay=1",
"player", "640", "390", "8", null, null, params, atts);

function onYouTubePlayerReady(playerId) {
 ytplayer = document.getElementById(
          "myytplayer");
 ytplayer.addEventListener(
          "onStateChange",
          "onPlayerStateChange");
}
```

Comparing the iframe API to the AS3 Player API

JavaScript Operations

Functionality	Example	Supported
Queueing functions	loadVideoById, cueVideoById	✓
Playback controls and player settings	seekTo, setVolume	✓
Playback status	getVideoBytesTotal	✓
Playback quality	getPlaybackQuality, setPlaybackQuality	✓
Retrieving video information	getDuration, getVideoEmbedCode	✓

Comparing the iframe API to the AS3 Player API

JavaScript Event Handling

```
function onYouTubePlayerAPIReady() {
  var player;
  player = new YT.Player('player', {
    width: 1280,
    height: 720,
    videoId: 'u1zgFlCw8Aw',
    events: {
      'onReady': onPlayerReady,
      'onPlaybackQualityChange': onPlayerPlaybackQualityChange,
      'onStateChange': onPlayerStateChange,
      'onError': onPlayerError
    }
  });
}
```

OR (AS3 player, registration also supported by iframe API)

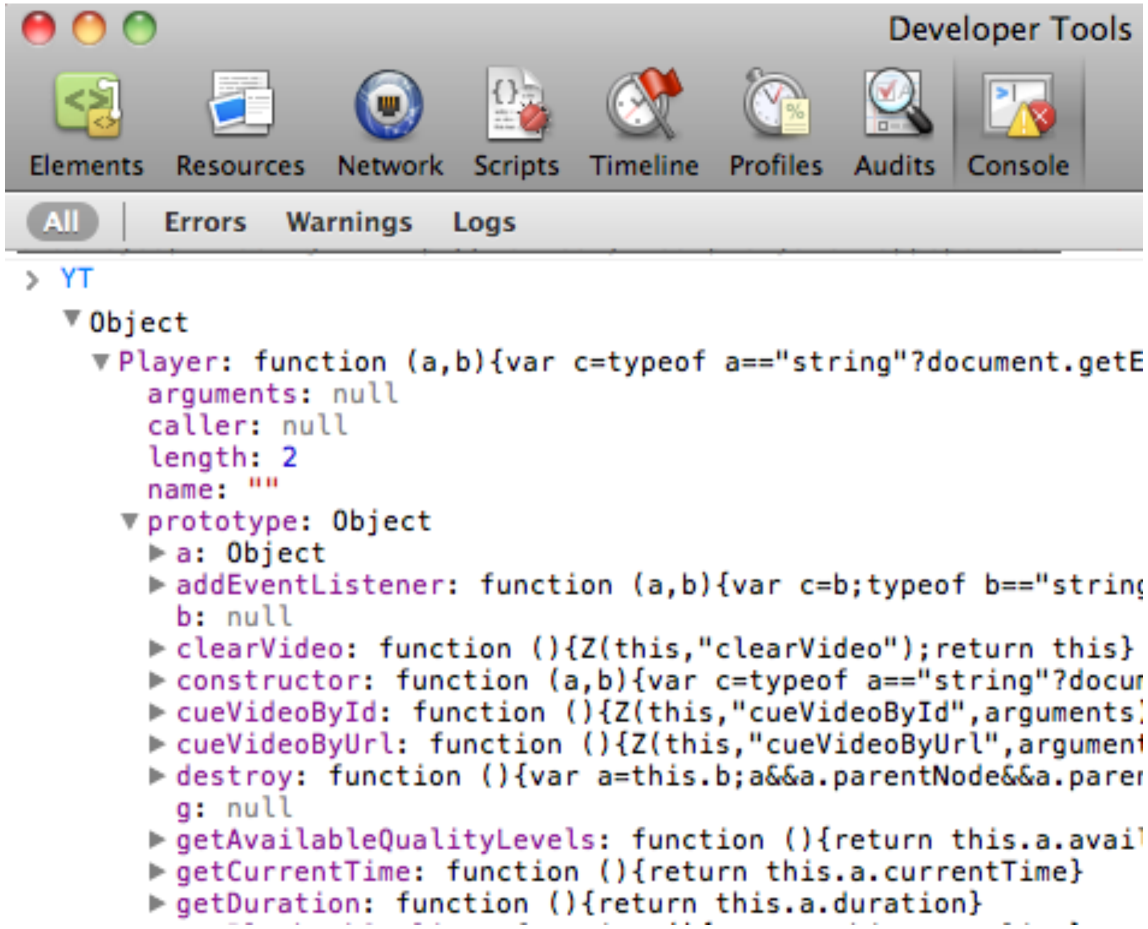
```
function onYouTubePlayerReady(playerId) {
  player = document.getElementById(playerId);
  player.addEventListener('onReady','onPlayerReady');
  player.addEventListener('onPlaybackQualityChange',
                          'onPlayerPlaybackQualityChange');
  player.addEventListener('onStateChange','onPlayerStateChange');
  player.addEventListener('onError','onPlayerError');
```

Note: YT.PlayerState.BUFFERING(3)not supported yet

Comparing the iframe API to the AS3 Player API

Explore API Using Chrome Dev Console

[Example]



```
> YT
  ▼Object
    ▼Player: function (a,b){var c=typeof a=="string"?document.getE
        arguments: null
        caller: null
        length: 2
        name: ""
      ▼prototype: Object
        ▶a: Object
        ▶addEventListener: function (a,b){var c=b;typeof b=="strin
         b: null
        ▶clearVideo: function (){Z(this,"clearVideo");return this}
        ▶constructor: function (a,b){var c=typeof a=="string"?docu
        ▶cueVideoById: function (){Z(this,"cueVideoById",arguments
        ▶cueVideoByUrl: function (){Z(this,"cueVideoByUrl",argumen
        ▶destroy: function (){var a=this.b;a&&a.parentNode&&a.pare
         g: null
        ▶getAvailableQualityLevels: function (){return this.a.avai
        ▶getCurrentTime: function (){return this.a.currentTime}
        ▶getDuration: function (){return this.a.duration}
```

Session Overview

- What is the iframe player?
- HTML5 playback in detail.
- Details of writing and exposing a JavaScript API for controlling the player.
- Comparing the iframe API to the AS3 player API.
- iframe player API example application.

iframe Player API Example Application

- Provides YouTube feed player functionality—think playlist player, but for any source of YouTube videos.
- HTML5 + JavaScript + CSS.
- Also powered by the YouTube Data API.
- Hopefully useful in its own right, but written to illustrate iframe Player API usage.

iframe Player API Example Application

HTML5

- <video> playback (for supported videos) via the iframe Player embed.
    - Using "chromeless" (controls=0) version of player.
- <svg> for player's Pause and Play buttons.
- <input type='range'> for Seek and Volume controls.
- Google Chrome currently supports all features.
    - Some other browsers offer a subset, i.e. no support for <input type='range'>

iframe Player API Example Application

JavaScript

- Lots of jQuery for plumbing!
  - Simplifies JSON-P access to the YouTube Data API.
  - Simplifies everything else as well.
- iframe Player API interaction is all JavaScript.
  - Responding to events, controlling playback, etc.

iframe Player API Example Application
CSS

- Basic CSS styling.
- Webfonts via the Google Font API.
- Bare bones design, but easy to change.

iframe Player API Example Application

Demo

http://gdata-samples.googlecode.com/svn/trunk/ytplayer/iframe/index.html

(or http://goo.gl/ncqF7)

iframe Player API Example Application

Handling Player Events

- A custom YouTube Player needs to understand various YouTube events and handle them appropriately.
  - onReady
  - onError
  - onStateChange
- onReady is fired when the Player is loaded and API methods can be called.
- onError is fired when a video can't be played.
- onStateChange is for "everything else".
  - YT.PlayerState.ENDED
  - YT.PlayerState.PLAYING
  - YT.PlayerState.PAUSED
  - YT.PlayerState.BUFFERING
  - YT.PlayerState.CUED

iframe Player API Example Application

Handling State Changes

- Recommended practice is to make no assumptions about global state or what triggered event.
  - You may think YT.PlayerState.PLAYING was triggered by your Play button, but it really was the Player itself.
  - Safer to explicitly set each UI element to the appropriate values (rather than toggling!) each time.

iframe Player API Example Application

Handling State Changes

```
function enable() {
  $.each(arguments, function(i, id) {
    $('#' + id).attr('disabled', false);
  });
}

function disable() {
  $.each(arguments, function(i, id) {
    $('#' + id).attr('disabled', true);
  });
}

function setSeekBarInterval() {
  seekBarInterval = setInterval(function() {
    var currentTime = Math.round(player.getCurrentTime());
    $('#currentTime').html(secondsToMmSs(currentTime));
    $('#seek').val(currentTime);
  }, 1000);
}
```

iframe Player API Example Application

Handling State Changes

```
case YT.PlayerState.CUED:
    enable('play');
    disable('pause', 'volume', 'seek');
break;

case YT.PlayerState.PAUSED:
    enable('play', 'volume', 'seek');
    disable('pause');
    if (seekBarInterval != null) {
        clearInterval(seekBarInterval);
        seekBarInterval = null;
    }
break;
```

iframe Player API Example Application
Handling State Changes

```
case YT.PlayerState.PLAYING:
    if (seekBarInterval != null) {
        clearInterval(seekBarInterval);
    }
    setSeekBarInterval();

    enable('pause', 'volume', 'seek');
    disable('play');

    $('#volume').val(player.getVolume());
    var duration = Math.round(player.getDuration());
    $('#duration').html(secondsToMmSs(duration));
    $('#seek').attr('max', duration);
break;
```

iframe Player API Example Application

Handling State Changes

```
case YT.PlayerState.ENDED:
   if (seekBarInterval != null) {
      clearInterval(seekBarInterval);
      seekBarInterval = null;
   }

   var duration = Math.round(player.getDuration());
   $('#currentTime').html(secondsToMmSs(duration));
   $('#seek').val(duration);

   enable('play');
   disable('pause', 'volume', 'seek');

   playNextVideo(player);
break;
```

Google™ I|O

Hashtags: #io2011 #YouTube
Feedback: http://goo.gl/fdY2L

Questions? Answers!