`video.js` shadowing

Tasks

1. Auto Setup	Scattered	2. FullScreen	Scattered
3. Playback Rate	Scattered	4. Mute	Scattered
5. WebKit	Scattered	6. OldWebKit	Scattered
7. Mozilla	Scattered	8. Microsoft	Scattered
9. BigPlay Button	Scattered	10. Loading Spinner	Scattered

Files Ordering

1. js/setup.js
2. js/big-play-button.js
3. js/control-bar/fullscreen-toggle.js
4. js/loading-spinner.js
5. js/control-bar/mute-toogle.js
6. js/control-bar/playback-rate-menu-button.js
7. js/fullscreen-api.js
8. js/exports.js
9. js/control-bar/control-bar.js
10. js/media/html5.js
11. js/player_externs.js
12. js/player.js
13. js/tracks.js
14. js/control-bar/volume-menu-button.js
15. js/media/flash.js
16. js/core.js

Files shadowing

js/setup.js

- **34 lines of code**

```
vjs.autoSetup = function(){
  var options, vid, player,
      vids = document.getElementsByTagName('video');
  if (vids && vids.length > 0) {
    for (var i=0,j=vids.length; i<j; i++) {
      vid = vids[i];
      if (vid && vid.getAttribute) {
        if (vid['player'] === undefined) {
          options = vid.getAttribute('data-setup');
          if (options !== null) {
```

```
            options = vjs.JSON.parse(options || '{}');
            player = videojs(vid, options);
          }
        }
      } else {
        vjs.autoSetupTimeout(1);
        break;
      }
    }
  } else if (!vjs.windowLoaded) {
    vjs.autoSetupTimeout(1);
  }
};
vjs.autoSetupTimeout = function(wait){
  setTimeout(vjs.autoSetup, wait);
};
if (document.readyState === 'complete') {
  vjs.windowLoaded = true;
} else {
  vjs.one(window, 'load', function(){
    vjs.windowLoaded = true;
  });
}
vjs.autoSetupTimeout(1);
```

js/big-play-button.js

- **11 lines of code**

```
vjs.BigPlayButton = vjs.Button.extend();
vjs.BigPlayButton.prototype.createEl = function(){
  return vjs.Button.prototype.createEl.call(this, 'div', {
    className: 'vjs-big-play-button',
    innerHTML: '<span aria-hidden="true"></span>',
    'aria-label': 'play video'
  });
};
vjs.BigPlayButton.prototype.onClick = function(){
  this.player_.play();
};
```

js/control-bar/fullscreen-toggle.js

- **18 lines of code**

```
vjs.FullscreenToggle = vjs.Button.extend({
  init: function(player, options){
    vjs.Button.call(this, player, options);
  }
});
vjs.FullscreenToggle.prototype.buttonText = 'Fullscreen';
vjs.FullscreenToggle.prototype.buildCSSClass = function(){
  return 'vjs-fullscreen-control ' + vjs.Button.prototype.buildCSSClass.call(this);
};
vjs.FullscreenToggle.prototype.onClick = function(){
  if (!this.player_.isFullscreen()) {
    this.player_.requestFullscreen();
    this.controlText_.innerHTML = 'Non-Fullscreen';
  } else {
    this.player_.exitFullscreen();
    this.controlText_.innerHTML = 'Fullscreen';
  }
};
```

js/loading-spinner.js

- **17 lines of code**

```
vjs.LoadingSpinner = vjs.Component.extend({
  init: function(player, options){
    vjs.Component.call(this, player, options);
    player.on('canplay', vjs.bind(this, this.hide));
    player.on('canplaythrough', vjs.bind(this, this.hide));
    player.on('playing', vjs.bind(this, this.hide));
```

```
    player.on('seeking', vjs.bind(this, this.show));
    player.on('seeked', vjs.bind(this, this.hide));
    player.on('ended', vjs.bind(this, this.hide));
    player.on('waiting', vjs.bind(this, this.show));
  }
});
vjs.LoadingSpinner.prototype.createEl = function(){
  return vjs.Component.prototype.createEl.call(this, 'div', {
    className: 'vjs-loading-spinner'
  });
};
```

js/control-bar/mute-toogle.js

- **49 lines of code**

```
vjs.MuteToggle = vjs.Button.extend({
  init: function(player, options){
    vjs.Button.call(this, player, options);
    player.on('volumechange', vjs.bind(this, this.update));
    if (player.tech && player.tech.features && player.tech.features['volumeControl']
=== false) {
      this.addClass('vjs-hidden');
    }
    player.on('loadstart', vjs.bind(this, function(){
      if (player.tech.features && player.tech.features['volumeControl'] === false) {
        this.addClass('vjs-hidden');
      } else {
        this.removeClass('vjs-hidden');
      }
    }));
  }
});
vjs.MuteToggle.prototype.createEl = function(){
  return vjs.Button.prototype.createEl.call(this, 'div', {
    className: 'vjs-mute-control vjs-control',
    innerHTML: '<div><span class="vjs-control-text">Mute</span></div>'
  });
};
vjs.MuteToggle.prototype.onClick = function(){
  this.player_.muted( this.player_.muted() ? false : true );
};
vjs.MuteToggle.prototype.update = function(){
  var vol = this.player_.volume(),
      level = 3;
  if (vol === 0 || this.player_.muted()) {
    level = 0;
  } else if (vol < 0.33) {
    level = 1;
  } else if (vol < 0.67) {
    level = 2;
  }
  if(this.player_.muted()){
      if(this.el_.children[0].children[0].innerHTML!='Unmute'){
          this.el_.children[0].children[0].innerHTML = 'Unmute'; // change the button
text to "Unmute"
      }
  } else {
      if(this.el_.children[0].children[0].innerHTML!='Mute'){
          this.el_.children[0].children[0].innerHTML = 'Mute'; // change the button
text to "Mute"
      }
  }
  for (var i = 0; i < 4; i++) {
    vjs.removeClass(this.el_, 'vjs-vol-'+i);
  }
  vjs.addClass(this.el_, 'vjs-vol-'+level);
};
```

js/control-bar/playback-rate-menu-button.js

- **86 lines of code**

```
vjs.PlaybackRateMenuButton = vjs.MenuButton.extend({
  init: function(player, options){
    vjs.MenuButton.call(this, player, options);
    this.updateVisibility();
    this.updateLabel();
    player.on('loadstart', vjs.bind(this, this.updateVisibility));
    player.on('ratechange', vjs.bind(this, this.updateLabel));
  }
});
vjs.PlaybackRateMenuButton.prototype.createEl = function(){
  var el = vjs.Component.prototype.createEl.call(this, 'div', {
    className: 'vjs-playback-rate vjs-menu-button vjs-control',
    innerHTML: '<div class="vjs-control-content"><span class="vjs-control-
text">Playback Rate</span></div>'
  });
  this.labelEl_ = vjs.createEl('div', {
    className: 'vjs-playback-rate-value',
    innerHTML: 1.0
  });
  el.appendChild(this.labelEl_);
  return el;
};
vjs.PlaybackRateMenuButton.prototype.createMenu = function(){
  var menu = new vjs.Menu(this.player());
  var rates = this.player().options()['playbackRates'];
  if (rates) {
    for (var i = rates.length - 1; i >= 0; i--) {
      menu.addChild(
        new vjs.PlaybackRateMenuItem(this.player(), { 'rate': rates[i] + 'x'})
        );
    };
  }

  return menu;
};
vjs.PlaybackRateMenuButton.prototype.updateARIAAttributes = function(){
  this.el().setAttribute('aria-valuenow', this.player().playbackRate());
};
vjs.PlaybackRateMenuButton.prototype.onClick = function(){
  var currentRate = this.player().playbackRate();
  var rates = this.player().options()['playbackRates'];
  var newRate = rates[0];
  for (var i = 0; i <rates.length ; i++) {
    if (rates[i] > currentRate) {
      newRate = rates[i];
      break;
    }
  };
  this.player().playbackRate(newRate);
};
vjs.PlaybackRateMenuButton.prototype.playbackRateSupported = function(){
  return this.player().tech
    && this.player().tech.features['playbackRate']
    && this.player().options()['playbackRates']
    && this.player().options()['playbackRates'].length > 0
  ;
};
vjs.PlaybackRateMenuButton.prototype.updateVisibility = function(){
  if (this.playbackRateSupported()) {
    this.removeClass('vjs-hidden');
  } else {
    this.addClass('vjs-hidden');
  }
};
vjs.PlaybackRateMenuButton.prototype.updateLabel = function(){
  if (this.playbackRateSupported()) {
    this.labelEl_.innerHTML = this.player().playbackRate() + 'x';
  }
};
vjs.PlaybackRateMenuItem = vjs.MenuItem.extend({
  contentElType: 'button',
  init: function(player, options){
    var label = this.label = options['rate'];
    var rate = this.rate = parseFloat(label, 10);
```

```
    options['label'] = label;
    options['selected'] = rate === 1;
    vjs.MenuItem.call(this, player, options);
    this.player().on('ratechange', vjs.bind(this, this.update));
  }
});
vjs.PlaybackRateMenuItem.prototype.onClick = function(){
  vjs.MenuItem.prototype.onClick.call(this);
  this.player().playbackRate(this.rate);
};
vjs.PlaybackRateMenuItem.prototype.update = function(){
  this.selected(this.player().playbackRate() == this.rate);
};
```

js/fullscreen-api.js

- **70 lines of code**

```
(function(){
  var apiMap, specApi, browserApi, i;
  vjs.browser.fullscreenAPI;
  apiMap = [
    [
      'requestFullscreen',
      'exitFullscreen',
      'fullscreenElement',
      'fullscreenEnabled',
      'fullscreenchange',
      'fullscreenerror'
    ]
    ,
    [
      'webkitRequestFullscreen',
      'webkitExitFullscreen',
      'webkitFullscreenElement',
      'webkitFullscreenEnabled',
      'webkitfullscreenchange',
      'webkitfullscreenerror'
    ]
    ,
    [
      'webkitRequestFullScreen',
      'webkitCancelFullScreen',
      'webkitCurrentFullScreenElement',
      'webkitCancelFullScreen',
      'webkitfullscreenchange',
      'webkitfullscreenerror'
    ]
    ,
    [
      'mozRequestFullScreen',
      'mozCancelFullScreen',
      'mozFullScreenElement',
      'mozFullScreenEnabled',
      'mozfullscreenchange',
      'mozfullscreenerror'
     ]
    ,[
      'msRequestFullscreen',
      'msExitFullscreen',
      'msFullscreenElement',
      'msFullscreenEnabled',
      'MSFullscreenChange',
      'MSFullscreenError'
    ]
  ];
  specApi = apiMap[0];
  for (i=0; i<apiMap.length; i++) {
    if (apiMap[i][1] in document) {
      browserApi = apiMap[i];
      break;
    }
```

```
  }
  if (browserApi) {
    vjs.browser.fullscreenAPI = {};
    for (i=0; i<browserApi.length; i++) {
      vjs.browser.fullscreenAPI[specApi[i]] = browserApi[i];
    }
  }
})();
```

js/exports.js

- **125 lines of code**

```
goog.exportSymbol('videojs', vjs);
goog.exportSymbol('_V_', vjs);
goog.exportSymbol('videojs.options', vjs.options);
goog.exportSymbol('videojs.players', vjs.players);
goog.exportSymbol('videojs.TOUCH_ENABLED', vjs.TOUCH_ENABLED);
goog.exportSymbol('videojs.cache', vjs.cache);
goog.exportSymbol('videojs.Component', vjs.Component);
goog.exportProperty(vjs.Component.prototype, 'player',
vjs.Component.prototype.player);
goog.exportProperty(vjs.Component.prototype, 'options',
vjs.Component.prototype.options);
goog.exportProperty(vjs.Component.prototype, 'init', vjs.Component.prototype.init);
goog.exportProperty(vjs.Component.prototype, 'dispose',
vjs.Component.prototype.dispose);
goog.exportProperty(vjs.Component.prototype, 'createEl',
vjs.Component.prototype.createEl);
goog.exportProperty(vjs.Component.prototype, 'contentEl',
vjs.Component.prototype.contentEl);
goog.exportProperty(vjs.Component.prototype, 'el', vjs.Component.prototype.el);
goog.exportProperty(vjs.Component.prototype, 'addChild',
vjs.Component.prototype.addChild);
goog.exportProperty(vjs.Component.prototype, 'getChild',
vjs.Component.prototype.getChild);
goog.exportProperty(vjs.Component.prototype, 'getChildById',
vjs.Component.prototype.getChildById);
goog.exportProperty(vjs.Component.prototype, 'children',
vjs.Component.prototype.children);
goog.exportProperty(vjs.Component.prototype, 'initChildren',
vjs.Component.prototype.initChildren);
goog.exportProperty(vjs.Component.prototype, 'removeChild',
vjs.Component.prototype.removeChild);
goog.exportProperty(vjs.Component.prototype, 'on', vjs.Component.prototype.on);
goog.exportProperty(vjs.Component.prototype, 'off', vjs.Component.prototype.off);
goog.exportProperty(vjs.Component.prototype, 'one', vjs.Component.prototype.one);
goog.exportProperty(vjs.Component.prototype, 'trigger',
vjs.Component.prototype.trigger);
goog.exportProperty(vjs.Component.prototype, 'triggerReady',
vjs.Component.prototype.triggerReady);
goog.exportProperty(vjs.Component.prototype, 'show', vjs.Component.prototype.show);
goog.exportProperty(vjs.Component.prototype, 'hide', vjs.Component.prototype.hide);
goog.exportProperty(vjs.Component.prototype, 'width', vjs.Component.prototype.width);
goog.exportProperty(vjs.Component.prototype, 'height',
vjs.Component.prototype.height);
goog.exportProperty(vjs.Component.prototype, 'dimensions',
vjs.Component.prototype.dimensions);
goog.exportProperty(vjs.Component.prototype, 'ready', vjs.Component.prototype.ready);
goog.exportProperty(vjs.Component.prototype, 'addClass',
vjs.Component.prototype.addClass);
goog.exportProperty(vjs.Component.prototype, 'removeClass',
vjs.Component.prototype.removeClass);
goog.exportProperty(vjs.Component.prototype, 'buildCSSClass',
vjs.Component.prototype.buildCSSClass);
goog.exportProperty(vjs.Player.prototype, 'ended', vjs.Player.prototype.ended);
goog.exportSymbol('videojs.MediaLoader', vjs.MediaLoader);
goog.exportSymbol('videojs.TextTrackDisplay', vjs.TextTrackDisplay);
goog.exportSymbol('videojs.ControlBar', vjs.ControlBar);
goog.exportSymbol('videojs.Button', vjs.Button);
goog.exportSymbol('videojs.PlayToggle', vjs.PlayToggle);
goog.exportSymbol('videojs.FullscreenToggle', vjs.FullscreenToggle);
goog.exportSymbol('videojs.BigPlayButton', vjs.BigPlayButton);
goog.exportSymbol('videojs.LoadingSpinner', vjs.LoadingSpinner);
goog.exportSymbol('videojs.CurrentTimeDisplay', vjs.CurrentTimeDisplay);
```

```
goog.exportSymbol('videojs.DurationDisplay', vjs.DurationDisplay);
goog.exportSymbol('videojs.TimeDivider', vjs.TimeDivider);
goog.exportSymbol('videojs.RemainingTimeDisplay', vjs.RemainingTimeDisplay);
goog.exportSymbol('videojs.LiveDisplay', vjs.LiveDisplay);
goog.exportSymbol('videojs.ErrorDisplay', vjs.ErrorDisplay);
goog.exportSymbol('videojs.Slider', vjs.Slider);
goog.exportSymbol('videojs.ProgressControl', vjs.ProgressControl);
goog.exportSymbol('videojs.SeekBar', vjs.SeekBar);
goog.exportSymbol('videojs.LoadProgressBar', vjs.LoadProgressBar);
goog.exportSymbol('videojs.PlayProgressBar', vjs.PlayProgressBar);
goog.exportSymbol('videojs.SeekHandle', vjs.SeekHandle);
goog.exportSymbol('videojs.VolumeControl', vjs.VolumeControl);
goog.exportSymbol('videojs.VolumeBar', vjs.VolumeBar);
goog.exportSymbol('videojs.VolumeLevel', vjs.VolumeLevel);
goog.exportSymbol('videojs.VolumeMenuButton', vjs.VolumeMenuButton);
goog.exportSymbol('videojs.VolumeHandle', vjs.VolumeHandle);
goog.exportSymbol('videojs.MuteToggle', vjs.MuteToggle);
goog.exportSymbol('videojs.PosterImage', vjs.PosterImage);
goog.exportSymbol('videojs.Menu', vjs.Menu);
goog.exportSymbol('videojs.MenuItem', vjs.MenuItem);
goog.exportSymbol('videojs.MenuButton', vjs.MenuButton);
goog.exportSymbol('videojs.PlaybackRateMenuButton', vjs.PlaybackRateMenuButton);
goog.exportProperty(vjs.MenuButton.prototype, 'createItems',
vjs.MenuButton.prototype.createItems);
goog.exportProperty(vjs.TextTrackButton.prototype, 'createItems',
vjs.TextTrackButton.prototype.createItems);
goog.exportProperty(vjs.ChaptersButton.prototype, 'createItems',
vjs.ChaptersButton.prototype.createItems);
goog.exportSymbol('videojs.SubtitlesButton', vjs.SubtitlesButton);
goog.exportSymbol('videojs.CaptionsButton', vjs.CaptionsButton);
goog.exportSymbol('videojs.ChaptersButton', vjs.ChaptersButton);
goog.exportSymbol('videojs.MediaTechController', vjs.MediaTechController);
goog.exportProperty(vjs.MediaTechController.prototype, 'features',
vjs.MediaTechController.prototype.features);
goog.exportProperty(vjs.MediaTechController.prototype.features, 'volumeControl',
vjs.MediaTechController.prototype.features.volumeControl);
goog.exportProperty(vjs.MediaTechController.prototype.features, 'fullscreenResize',
vjs.MediaTechController.prototype.features.fullscreenResize);
goog.exportProperty(vjs.MediaTechController.prototype.features, 'progressEvents',
vjs.MediaTechController.prototype.features.progressEvents);
goog.exportProperty(vjs.MediaTechController.prototype.features, 'timeupdateEvents',
vjs.MediaTechController.prototype.features.timeupdateEvents);
goog.exportProperty(vjs.MediaTechController.prototype, 'setPoster',
vjs.MediaTechController.prototype.setPoster);
goog.exportSymbol('videojs.Html5', vjs.Html5);
goog.exportProperty(vjs.Html5, 'Events', vjs.Html5.Events);
goog.exportProperty(vjs.Html5, 'isSupported', vjs.Html5.isSupported);
goog.exportProperty(vjs.Html5, 'canPlaySource', vjs.Html5.canPlaySource);
goog.exportProperty(vjs.Html5, 'patchCanPlayType', vjs.Html5.patchCanPlayType);
goog.exportProperty(vjs.Html5, 'unpatchCanPlayType', vjs.Html5.unpatchCanPlayType);
goog.exportProperty(vjs.Html5.prototype, 'setCurrentTime',
vjs.Html5.prototype.setCurrentTime);
goog.exportProperty(vjs.Html5.prototype, 'setVolume', vjs.Html5.prototype.setVolume);
goog.exportProperty(vjs.Html5.prototype, 'setMuted', vjs.Html5.prototype.setMuted);
goog.exportProperty(vjs.Html5.prototype, 'setPreload',
vjs.Html5.prototype.setPreload);
goog.exportProperty(vjs.Html5.prototype, 'setAutoplay',
vjs.Html5.prototype.setAutoplay);
goog.exportProperty(vjs.Html5.prototype, 'setLoop', vjs.Html5.prototype.setLoop);
goog.exportProperty(vjs.Html5.prototype, 'enterFullScreen',
vjs.Html5.prototype.enterFullScreen);
goog.exportProperty(vjs.Html5.prototype, 'exitFullScreen',
vjs.Html5.prototype.exitFullScreen);
goog.exportProperty(vjs.Html5.prototype, 'playbackRate',
vjs.Html5.prototype.playbackRate);
goog.exportProperty(vjs.Html5.prototype, 'setPlaybackRate',
vjs.Html5.prototype.setPlaybackRate);
goog.exportSymbol('videojs.Flash', vjs.Flash);
goog.exportProperty(vjs.Flash, 'isSupported', vjs.Flash.isSupported);
goog.exportProperty(vjs.Flash, 'canPlaySource', vjs.Flash.canPlaySource);
goog.exportProperty(vjs.Flash, 'onReady', vjs.Flash['onReady']);
goog.exportSymbol('videojs.TextTrack', vjs.TextTrack);
goog.exportProperty(vjs.TextTrack.prototype, 'label', vjs.TextTrack.prototype.label);
goog.exportProperty(vjs.TextTrack.prototype, 'kind', vjs.TextTrack.prototype.kind);
goog.exportProperty(vjs.TextTrack.prototype, 'mode', vjs.TextTrack.prototype.mode);
```

```
goog.exportProperty(vjs.TextTrack.prototype, 'cues', vjs.TextTrack.prototype.cues);
goog.exportProperty(vjs.TextTrack.prototype, 'activeCues',
vjs.TextTrack.prototype.activeCues);
goog.exportSymbol('videojs.CaptionsTrack', vjs.CaptionsTrack);
goog.exportSymbol('videojs.SubtitlesTrack', vjs.SubtitlesTrack);
goog.exportSymbol('videojs.ChaptersTrack', vjs.ChaptersTrack);
goog.exportSymbol('videojs.autoSetup', vjs.autoSetup);
goog.exportSymbol('videojs.plugin', vjs.plugin);
goog.exportSymbol('videojs.createTimeRange', vjs.createTimeRange);
goog.exportSymbol('videojs.util', vjs.util);
goog.exportProperty(vjs.util, 'mergeOptions', vjs.util.mergeOptions);
```

js/control-bar/control-bar.js

- **29 lines of code**

```
vjs.ControlBar = vjs.Component.extend();
vjs.ControlBar.prototype.options_ = {
  loadEvent: 'play',
  children: {
    'playToggle': {},
    'currentTimeDisplay': {},
    'timeDivider': {},
    'durationDisplay': {},
    'remainingTimeDisplay': {},
    'liveDisplay': {},
    'progressControl': {},
    'fullscreenToggle': {},
    'volumeControl': {},
    'muteToggle': {},
    // 'volumeMenuButton': {},
    'playbackRateMenuButton': {}
  }
};
vjs.ControlBar.prototype.createEl = function(){
  return vjs.createEl('div', {
    className: 'vjs-control-bar'
  });
};
```

js/media/html5.js

- **238 lines of code**

```
vjs.Html5 = vjs.MediaTechController.extend({
  init: function(player, options, ready){
    this.features['volumeControl'] = vjs.Html5.canControlVolume();
    this.features['playbackRate'] = vjs.Html5.canControlPlaybackRate();
    this.features['movingMediaElementInDOM'] = !vjs.IS_IOS;
    this.features['fullscreenResize'] = true;
    vjs.MediaTechController.call(this, player, options, ready);
    this.setupTriggers();
    var source = options['source'];
    if (source && this.el_.currentSrc === source.src && this.el_.networkState > 0) {
      player.ready(function(){
        player.trigger('loadstart');
      });
    } else if (source) {
      this.el_.src = source.src;
    }
    if (vjs.TOUCH_ENABLED && player.options()['nativeControlsForTouch'] !== false) {
      this.useNativeControls();
    }
    player.ready(function(){
      if (this.tag && this.options_['autoplay'] && this.paused()) {
        delete this.tag['poster']; // Chrome Fix. Fixed in Chrome v16.
        this.play();
      }
    });
    this.triggerReady();
  }
});
vjs.Html5.prototype.dispose = function(){
  vjs.MediaTechController.prototype.dispose.call(this);
```

```
};
vjs.Html5.prototype.createEl = function(){
  var player = this.player_,
      el = player.tag,
      newEl,
      clone;
  if (!el || this.features['movingMediaElementInDOM'] === false) {
    if (el) {
      clone = el.cloneNode(false);
      vjs.Html5.disposeMediaElement(el);
      el = clone;
      player.tag = null;
    } else {
      el = vjs.createEl('video', {
        id:player.id() + '_html5_api',
        className:'vjs-tech'
      });
    }
    el['player'] = player;
    vjs.insertFirst(el, player.el());
  }
  var attrs = ['autoplay','preload','loop','muted'];
  for (var i = attrs.length - 1; i >= 0; i--) {
    var attr = attrs[i];
    if (player.options_[attr] !== null) {
      el[attr] = player.options_[attr];
    }
  }
  return el;
};
vjs.Html5.prototype.setupTriggers = function(){
  for (var i = vjs.Html5.Events.length - 1; i >= 0; i--) {
    vjs.on(this.el_, vjs.Html5.Events[i], vjs.bind(this, this.eventHandler));
  }
};
vjs.Html5.prototype.eventHandler = function(evt){
  if (evt.type == 'error') {
    this.player().error(this.error().code);
  } else {
    evt.bubbles = false;
    this.player().trigger(evt);
  }
};
vjs.Html5.prototype.useNativeControls = function(){
  var tech, player, controlsOn, controlsOff, cleanUp;
  tech = this;
  player = this.player();
  tech.setControls(player.controls());
  controlsOn = function(){
    tech.setControls(true);
  };
  controlsOff = function(){
    tech.setControls(false);
  };
  player.on('controlsenabled', controlsOn);
  player.on('controlsdisabled', controlsOff);
  cleanUp = function(){
    player.off('controlsenabled', controlsOn);
    player.off('controlsdisabled', controlsOff);
  };
  tech.on('dispose', cleanUp);
  player.on('usingcustomcontrols', cleanUp);
  player.usingNativeControls(true);
};
vjs.Html5.prototype.play = function(){ this.el_.play(); };
vjs.Html5.prototype.pause = function(){ this.el_.pause(); };
vjs.Html5.prototype.paused = function(){ return this.el_.paused; };
vjs.Html5.prototype.currentTime = function(){ return this.el_.currentTime; };
vjs.Html5.prototype.setCurrentTime = function(seconds){
  try {
    this.el_.currentTime = seconds;
  } catch(e) {
    vjs.log(e, 'Video is not ready. (Video.js)');
  }
```

```
};
vjs.Html5.prototype.duration = function(){ return this.el_.duration || 0; };
vjs.Html5.prototype.buffered = function(){ return this.el_.buffered; };
vjs.Html5.prototype.volume = function(){ return this.el_.volume; };
vjs.Html5.prototype.setVolume = function(percentAsDecimal){ this.el_.volume =
percentAsDecimal; };
vjs.Html5.prototype.muted = function(){ return this.el_.muted; };
vjs.Html5.prototype.setMuted = function(muted){ this.el_.muted = muted; };
vjs.Html5.prototype.width = function(){ return this.el_.offsetWidth; };
vjs.Html5.prototype.height = function(){ return this.el_.offsetHeight; };
vjs.Html5.prototype.supportsFullScreen = function(){
  if (typeof this.el_.webkitEnterFullScreen == 'function') {
    if (/Android/.test(vjs.USER_AGENT) || !/Chrome|Mac OS X
10.5/.test(vjs.USER_AGENT)) {
      return true;
    }
  }
  return false;
};
vjs.Html5.prototype.enterFullScreen = function(){
  var video = this.el_;
  if (video.paused && video.networkState <= video.HAVE_METADATA) {
    this.el_.play();
    setTimeout(function(){
      video.pause();
      video.webkitEnterFullScreen();
    }, 0);
  } else {
    video.webkitEnterFullScreen();
  }
};
vjs.Html5.prototype.exitFullScreen = function(){
  this.el_.webkitExitFullScreen();
};
vjs.Html5.prototype.src = function(src){ this.el_.src = src; };
vjs.Html5.prototype.load = function(){ this.el_.load(); };
vjs.Html5.prototype.currentSrc = function(){ return this.el_.currentSrc; };
vjs.Html5.prototype.poster = function(){ return this.el_.poster; };
vjs.Html5.prototype.setPoster = function(val){ this.el_.poster = val; };
vjs.Html5.prototype.preload = function(){ return this.el_.preload; };
vjs.Html5.prototype.setPreload = function(val){ this.el_.preload = val; };
vjs.Html5.prototype.autoplay = function(){ return this.el_.autoplay; };
vjs.Html5.prototype.setAutoplay = function(val){ this.el_.autoplay = val; };
vjs.Html5.prototype.controls = function(){ return this.el_.controls; };
vjs.Html5.prototype.setControls = function(val){ this.el_.controls = !!val; };
vjs.Html5.prototype.loop = function(){ return this.el_.loop; };
vjs.Html5.prototype.setLoop = function(val){ this.el_.loop = val; };
vjs.Html5.prototype.error = function(){ return this.el_.error; };
vjs.Html5.prototype.seeking = function(){ return this.el_.seeking; };
vjs.Html5.prototype.ended = function(){ return this.el_.ended; };
vjs.Html5.prototype.defaultMuted = function(){ return this.el_.defaultMuted; };
vjs.Html5.prototype.playbackRate = function(){ return this.el_.playbackRate; };
vjs.Html5.prototype.setPlaybackRate = function(val){ this.el_.playbackRate = val; };
vjs.Html5.isSupported = function(){
  try {
    vjs.TEST_VID['volume'] = 0.5;
  } catch (e) {
    return false;
  }
  return !!vjs.TEST_VID.canPlayType;
};
vjs.Html5.canPlaySource = function(srcObj){
  try {
    return !!vjs.TEST_VID.canPlayType(srcObj.type);
  } catch(e) {
    return '';
  }
};
vjs.Html5.canControlVolume = function(){
  var volume =  vjs.TEST_VID.volume;
  vjs.TEST_VID.volume = (volume / 2) + 0.1;
  return volume !== vjs.TEST_VID.volume;
};
vjs.Html5.canControlPlaybackRate = function(){
```

```
  var playbackRate =  vjs.TEST_VID.playbackRate;
  vjs.TEST_VID.playbackRate = (playbackRate / 2) + 0.1;
  return playbackRate !== vjs.TEST_VID.playbackRate;
};
(function() {
  var canPlayType,
      mpegurlRE = /^application\/(?:x-|vnd\.apple\.)mpegurl/i,
      mp4RE = /^video\/mp4/i;
  vjs.Html5.patchCanPlayType = function() {
    if (vjs.ANDROID_VERSION >= 4.0) {
      if (!canPlayType) {
        canPlayType = vjs.TEST_VID.constructor.prototype.canPlayType;
      }
      vjs.TEST_VID.constructor.prototype.canPlayType = function(type) {
        if (type && mpegurlRE.test(type)) {
          return 'maybe';
        }
        return canPlayType.call(this, type);
      };
    }
    if (vjs.IS_OLD_ANDROID) {
      if (!canPlayType) {
        canPlayType = vjs.TEST_VID.constructor.prototype.canPlayType;
      }
      vjs.TEST_VID.constructor.prototype.canPlayType = function(type){
        if (type && mp4RE.test(type)) {
          return 'maybe';
        }
        return canPlayType.call(this, type);
      };
    }
  };
  vjs.Html5.unpatchCanPlayType = function() {
    var r = vjs.TEST_VID.constructor.prototype.canPlayType;
    vjs.TEST_VID.constructor.prototype.canPlayType = canPlayType;
    canPlayType = null;
    return r;
  };
  vjs.Html5.patchCanPlayType();
})();
vjs.Html5.Events =
'loadstart,suspend,abort,error,emptied,stalled,loadedmetadata,loadeddata,canplay,canpl
aythrough,playing,waiting,seeking,seeked,ended,durationchange,timeupdate,progress,play
,pause,ratechange,volumechange'.split(',');
vjs.Html5.disposeMediaElement = function(el){
  if (!el) { return; }
  el['player'] = null;
  if (el.parentNode) {
    el.parentNode.removeChild(el);
  }
  while(el.hasChildNodes()) {
    el.removeChild(el.firstChild);
  }
  el.removeAttribute('src');
  if (typeof el.load === 'function') {
    (function() {
      try {
        el.load();
      } catch (e) {
      }
    })();
  }
};
```

js/player_externs.js

- **45 lines of code**

```
videojs.Player = function(){};
videojs.Player.prototype.error = function(){};
videojs.Player.prototype.src = function(){};
videojs.Player.prototype.currentSrc = function(){};
videojs.Player.prototype.networkState = function(){};
videojs.Player.prototype.buffered = function(){};
```

```
videojs.Player.prototype.load = function(){};
videojs.Player.prototype.canPlayType = function(){};
videojs.Player.prototype.readyState = function(){};
videojs.Player.prototype.seeking = function(){};
videojs.Player.prototype.currentTime = function(){};
videojs.Player.prototype.startTime = function(){};
videojs.Player.prototype.duration = function(){};
videojs.Player.prototype.paused = function(){};
videojs.Player.prototype.defaultPlaybackRate = function(){};
videojs.Player.prototype.playbackRate = function(){};
videojs.Player.prototype.played = function(){};
videojs.Player.prototype.seekable = function(){};
videojs.Player.prototype.ended = function(){};
videojs.Player.prototype.autoplay = function(){};
videojs.Player.prototype.loop = function(){};
videojs.Player.prototype.play = function() {};
videojs.Player.prototype.pause = function() {};
videojs.Player.prototype.controls = function(){};
videojs.Player.prototype.volume = function(){};
videojs.Player.prototype.muted = function(){};
videojs.Player.prototype.width = function(){};
videojs.Player.prototype.height = function(){};
videojs.Player.prototype.videoWidth = function(){};
videojs.Player.prototype.videoHeight = function(){};
videojs.Player.prototype.poster = function(){};
videojs.Player.prototype.isFullscreen = function(){};
videojs.Player.prototype.isFullScreen = function(){}; /* deprecated */
videojs.Player.prototype.requestFullscreen = function(){};
videojs.Player.prototype.requestFullScreen = function(){}; /* deprecated */
videojs.Player.prototype.exitFullscreen = function(){};
videojs.Player.prototype.cancelFullScreen = function(){}; /* deprecated */
videojs.Player.prototype.textTracks = function(){};
videojs.Player.prototype.dispose = function(){};
videojs.Player.prototype.bufferedPercent = function(){};
videojs.Player.prototype.reportUserActivity = function(){};
videojs.Player.prototype.userActive = function(){};
videojs.Player.prototype.usingNativeControls = function(){};
```

js/player.js

- **708 lines of code**

```
vjs.Player = vjs.Component.extend({
  init: function(tag, options, ready){
    this.tag = tag; // Store the original tag used to set options
    tag.id = tag.id || 'vjs_video_' + vjs.guid++;
    options = vjs.obj.merge(this.getTagSettings(tag), options);
    this.cache_ = {};
    this.poster_ = options['poster'];
    this.controls_ = options['controls'];
    tag.controls = false;
    options.reportTouchActivity = false;
    this.ready(function(){
      this.on('loadstart', this.onLoadStart);
      this.on('ended', this.onEnded);
      this.on('play', this.onPlay);
      this.on('firstplay', this.onFirstPlay);
      this.on('pause', this.onPause);
      this.on('progress', this.onProgress);
      this.on('durationchange', this.onDurationChange);
      this.on('fullscreenchange', this.onFullscreenChange);
      });
    vjs.Component.call(this, this, options, ready);
    if (this.controls()) {
      this.addClass('vjs-controls-enabled');
    } else {
      this.addClass('vjs-controls-disabled');
    }
    vjs.players[this.id_] = this;
    if (options['plugins']) {
      vjs.obj.each(options['plugins'], function(key, val){
        this[key](val);
      }, this);
    }
```

```
    this.listenForUserActivity();
  }
});
vjs.Player.prototype.options_ = vjs.options;
vjs.Player.prototype.dispose = function(){
  this.trigger('dispose');
  this.off('dispose');
  vjs.players[this.id_] = null;
  if (this.tag && this.tag['player']) { this.tag['player'] = null; }
  if (this.el_ && this.el_['player']) { this.el_['player'] = null; }
  this.stopTrackingProgress();
  this.stopTrackingCurrentTime();
  if (this.tech) { this.tech.dispose(); }
  vjs.Component.prototype.dispose.call(this);
};
vjs.Player.prototype.getTagSettings = function(tag){
  var options = {
    'sources': [],
    'tracks': []
  };
  vjs.obj.merge(options, vjs.getAttributeValues(tag));
  if (tag.hasChildNodes()) {
    var children, child, childName, i, j;
    children = tag.childNodes;
    for (i=0,j=children.length; i<j; i++) {
      child = children[i];
      childName = child.nodeName.toLowerCase();
      if (childName === 'source') {
        options['sources'].push(vjs.getAttributeValues(child));
      } else if (childName === 'track') {
        options['tracks'].push(vjs.getAttributeValues(child));
      }
    }
  }
  return options;
};
vjs.Player.prototype.createEl = function(){
  var el = this.el_ = vjs.Component.prototype.createEl.call(this, 'div');
  var tag = this.tag;
  tag.removeAttribute('width');
  tag.removeAttribute('height');
  if (tag.hasChildNodes()) {
    var nodes, nodesLength, i, node, nodeName, removeNodes;
    nodes = tag.childNodes;
    nodesLength = nodes.length;
    removeNodes = [];
    while (nodesLength--) {
      node = nodes[nodesLength];
      nodeName = node.nodeName.toLowerCase();
      if (nodeName === 'track') {
        removeNodes.push(node);
      }
    }
    for (i=0; i<removeNodes.length; i++) {
      tag.removeChild(removeNodes[i]);
    }
  }
  el.id = tag.id;
  el.className = tag.className;
  tag.id += '_html5_api';
  tag.className = 'vjs-tech';
  tag['player'] = el['player'] = this;
  this.addClass('vjs-paused');
  this.width(this.options_['width'], true); // (true) Skip resize listener on load
  this.height(this.options_['height'], true);
  if (tag.parentNode) {
    tag.parentNode.insertBefore(el, tag);
  }
  vjs.insertFirst(tag, el); // Breaks iPhone, fixed in HTML5 setup.
  return el;
};
vjs.Player.prototype.loadTech = function(techName, source){
  if (this.tech) {
    this.unloadTech();
```

```
  }
  if (techName !== 'Html5' && this.tag) {
    vjs.Html5.disposeMediaElement(this.tag);
    this.tag = null;
  }
  this.techName = techName;
  this.isReady_ = false;
  var techReady = function(){
    this.player_.triggerReady();
    if (!this.features['progressEvents']) {
      this.player_.manualProgressOn();
    }
    if (!this.features['timeupdateEvents']) {
      this.player_.manualTimeUpdatesOn();
    }
  };
  var techOptions = vjs.obj.merge({ 'source': source, 'parentEl': this.el_ },
this.options_[techName.toLowerCase()]);
  if (source) {
    if (source.src == this.cache_.src && this.cache_.currentTime > 0) {
      techOptions['startTime'] = this.cache_.currentTime;
    }
    this.cache_.src = source.src;
  }
  this.tech = new window['videojs'][techName](this, techOptions);
  this.tech.ready(techReady);
};
vjs.Player.prototype.unloadTech = function(){
  this.isReady_ = false;
  this.tech.dispose();
  if (this.manualProgress) { this.manualProgressOff(); }
  if (this.manualTimeUpdates) { this.manualTimeUpdatesOff(); }
  this.tech = false;
};
vjs.Player.prototype.manualProgressOn = function(){
  this.manualProgress = true;
  this.trackProgress();
  if (this.tech) {
    this.tech.one('progress', function(){
      this.features['progressEvents'] = true;
      this.player_.manualProgressOff();
    });
  }
};
vjs.Player.prototype.manualProgressOff = function(){
  this.manualProgress = false;
  this.stopTrackingProgress();
};
vjs.Player.prototype.trackProgress = function(){
  this.progressInterval = setInterval(vjs.bind(this, function(){
    if (this.cache_.bufferEnd < this.buffered().end(0)) {
      this.trigger('progress');
    } else if (this.bufferedPercent() == 1) {
      this.stopTrackingProgress();
      this.trigger('progress'); // Last update
    }
  }), 500);
};
vjs.Player.prototype.stopTrackingProgress = function(){
clearInterval(this.progressInterval); };
vjs.Player.prototype.manualTimeUpdatesOn = function(){
  this.manualTimeUpdates = true;
  this.on('play', this.trackCurrentTime);
  this.on('pause', this.stopTrackingCurrentTime);
  if (this.tech) {
    this.tech.one('timeupdate', function(){
      this.features['timeupdateEvents'] = true;
      this.player_.manualTimeUpdatesOff();
    });
  }
};
vjs.Player.prototype.manualTimeUpdatesOff = function(){
  this.manualTimeUpdates = false;
  this.stopTrackingCurrentTime();
```

```
  this.off('play', this.trackCurrentTime);
  this.off('pause', this.stopTrackingCurrentTime);
};
vjs.Player.prototype.trackCurrentTime = function(){
  if (this.currentTimeInterval) { this.stopTrackingCurrentTime(); }
  this.currentTimeInterval = setInterval(vjs.bind(this, function(){
    this.trigger('timeupdate');
  }), 250); // 42 = 24 fps // 250 is what Webkit uses // FF uses 15
};
vjs.Player.prototype.stopTrackingCurrentTime = function(){
  clearInterval(this.currentTimeInterval);
  this.trigger('timeupdate');
};
vjs.Player.prototype.onLoadStart = function() {
  this.off('play', initFirstPlay);
  this.one('play', initFirstPlay);
  if (this.error()) {
    this.error(null);
  }
  vjs.removeClass(this.el_, 'vjs-has-started');
};
function initFirstPlay(e) {
  var fpEvent = { type: 'firstplay', target: this.el_ };
  var keepGoing = vjs.trigger(this.el_, fpEvent);
  if (!keepGoing) {
    e.preventDefault();
    e.stopPropagation();
    e.stopImmediatePropagation();
  }
}
vjs.Player.prototype.onLoadedMetaData;
vjs.Player.prototype.onLoadedData;
vjs.Player.prototype.onLoadedAllData;
vjs.Player.prototype.onPlay = function(){
  vjs.removeClass(this.el_, 'vjs-paused');
  vjs.addClass(this.el_, 'vjs-playing');
};
vjs.Player.prototype.onFirstPlay = function(){
    if(this.options_['starttime']){
      this.currentTime(this.options_['starttime']);
    }
    this.addClass('vjs-has-started');
};
vjs.Player.prototype.onPause = function(){
  vjs.removeClass(this.el_, 'vjs-playing');
  vjs.addClass(this.el_, 'vjs-paused');
};
vjs.Player.prototype.onTimeUpdate;
vjs.Player.prototype.onProgress = function(){
  if (this.bufferedPercent() == 1) {
    this.trigger('loadedalldata');
  }
};
vjs.Player.prototype.onEnded = function(){
  if (this.options_['loop']) {
    this.currentTime(0);
    this.play();
  }
};
vjs.Player.prototype.onDurationChange = function(){
  var duration = this.techGet('duration');
  if (duration) {
    if (duration < 0) {
      duration = Infinity;
    }
    this.duration(duration);
    if (duration === Infinity) {
      this.addClass('vjs-live');
    } else {
      this.removeClass('vjs-live');
    }
  }
};
vjs.Player.prototype.onVolumeChange;
```

```
vjs.Player.prototype.onFullscreenChange = function() {
  if (this.isFullscreen()) {
    this.addClass('vjs-fullscreen');
  } else {
    this.removeClass('vjs-fullscreen');
  }
};
vjs.Player.prototype.cache_;
vjs.Player.prototype.getCache = function(){
  return this.cache_;
};
vjs.Player.prototype.techCall = function(method, arg){
  if (this.tech && !this.tech.isReady_) {
    this.tech.ready(function(){
      this[method](arg);
    });
  } else {
    try {
      this.tech[method](arg);
    } catch(e) {
      vjs.log(e);
      throw e;
    }
  }
};
vjs.Player.prototype.techGet = function(method){
  if (this.tech && this.tech.isReady_) {
    try {
      return this.tech[method]();
    } catch(e) {
      if (this.tech[method] === undefined) {
        vjs.log('Video.js: ' + method + ' method not defined for '+this.techName+'
playback technology.', e);
      } else {
        if (e.name == 'TypeError') {
          vjs.log('Video.js: ' + method + ' unavailable on '+this.techName+' playback
technology element.', e);
          this.tech.isReady_ = false;
        } else {
          vjs.log(e);
        }
      }
      throw e;
    }
  }
  return;
};
vjs.Player.prototype.play = function(){
  this.techCall('play');
  return this;
};
vjs.Player.prototype.pause = function(){
  this.techCall('pause');
  return this;
};
vjs.Player.prototype.paused = function(){
  return (this.techGet('paused') === false) ? false : true;
};
vjs.Player.prototype.currentTime = function(seconds){
  if (seconds !== undefined) {
    this.techCall('setCurrentTime', seconds);
    if (this.manualTimeUpdates) { this.trigger('timeupdate'); }
    return this;
  }
  return this.cache_.currentTime = (this.techGet('currentTime') || 0);
};
vjs.Player.prototype.duration = function(seconds){
  if (seconds !== undefined) {
    this.cache_.duration = parseFloat(seconds);
    return this;
  }
  if (this.cache_.duration === undefined) {
    this.onDurationChange();
  }
```

```
  return this.cache_.duration || 0;
};
vjs.Player.prototype.remainingTime = function(){
  return this.duration() - this.currentTime();
};
vjs.Player.prototype.buffered = function(){
  var buffered = this.techGet('buffered'),
      start = 0,
      buflast = buffered.length - 1,
      end = this.cache_.bufferEnd = this.cache_.bufferEnd || 0;
  if (buffered && buflast >= 0 && buffered.end(buflast) !== end) {
    end = buffered.end(buflast);
    this.cache_.bufferEnd = end;
  }
  return vjs.createTimeRange(start, end);
};
vjs.Player.prototype.bufferedPercent = function(){
  return (this.duration()) ? this.buffered().end(0) / this.duration() : 0;
};
vjs.Player.prototype.volume = function(percentAsDecimal){
  var vol;
  if (percentAsDecimal !== undefined) {
    vol = Math.max(0, Math.min(1, parseFloat(percentAsDecimal))); // Force value to
between 0 and 1
    this.cache_.volume = vol;
    this.techCall('setVolume', vol);
    vjs.setLocalStorage('volume', vol);
    return this;
  }
  vol = parseFloat(this.techGet('volume'));
  return (isNaN(vol)) ? 1 : vol;
};
vjs.Player.prototype.muted = function(muted){
  if (muted !== undefined) {
    this.techCall('setMuted', muted);
    return this;
  }
  return this.techGet('muted') || false; // Default to false
};
vjs.Player.prototype.supportsFullScreen = function(){
  return this.techGet('supportsFullScreen') || false;
};
vjs.Player.prototype.isFullscreen_ = false;
vjs.Player.prototype.isFullscreen = function(isFS){
  if (isFS !== undefined) {
    this.isFullscreen_ = !!isFS;
    return this;
  }
  return this.isFullscreen_;
};
vjs.Player.prototype.isFullScreen = function(isFS){
  vjs.log.warn('player.isFullScreen() has been deprecated, use player.isFullscreen()
with a lowercase "s")');
  return this.isFullscreen(isFS);
};
vjs.Player.prototype.requestFullscreen = function(){
  var fsApi = vjs.browser.fullscreenAPI;
  this.isFullscreen(true);
  if (fsApi) {
    vjs.on(document, fsApi.fullscreenchange, vjs.bind(this, function(e){
      this.isFullscreen(document[fsApi.fullscreenElement]);
      if (this.isFullscreen() === false) {
        vjs.off(document, fsApi.fullscreenchange, arguments.callee);
      }
      this.trigger('fullscreenchange');
    }));
    this.el_[fsApi.requestFullscreen]();
  } else if (this.tech.supportsFullScreen()) {
    this.techCall('enterFullScreen');
  } else {
    this.enterFullWindow();
    this.trigger('fullscreenchange');
  }
  return this;
```

```
};
vjs.Player.prototype.requestFullScreen = function(){
  vjs.log.warn('player.requestFullScreen() has been deprecated, use
player.requestFullscreen() with a lowercase "s")');
  return this.requestFullscreen();
};
vjs.Player.prototype.exitFullscreen = function(){
  var fsApi = vjs.browser.fullscreenAPI;
  this.isFullscreen(false);
  if (fsApi) {
    document[fsApi.exitFullscreen]();
  } else if (this.tech.supportsFullScreen()) {
   this.techCall('exitFullScreen');
  } else {
   this.exitFullWindow();
   this.trigger('fullscreenchange');
  }
  return this;
};
vjs.Player.prototype.cancelFullScreen = function(){
  vjs.log.warn('player.cancelFullScreen() has been deprecated, use
player.exitFullscreen()');
  return this.exitFullscreen();
};
vjs.Player.prototype.enterFullWindow = function(){
  this.isFullWindow = true;
  this.docOrigOverflow = document.documentElement.style.overflow;
  vjs.on(document, 'keydown', vjs.bind(this, this.fullWindowOnEscKey));
  document.documentElement.style.overflow = 'hidden';
  vjs.addClass(document.body, 'vjs-full-window');
  this.trigger('enterFullWindow');
};
vjs.Player.prototype.fullWindowOnEscKey = function(event){
  if (event.keyCode === 27) {
    if (this.isFullscreen() === true) {
      this.exitFullscreen();
    } else {
      this.exitFullWindow();
    }
  }
};
vjs.Player.prototype.exitFullWindow = function(){
  this.isFullWindow = false;
  vjs.off(document, 'keydown', this.fullWindowOnEscKey);
  document.documentElement.style.overflow = this.docOrigOverflow;
  vjs.removeClass(document.body, 'vjs-full-window');
  this.trigger('exitFullWindow');
};
vjs.Player.prototype.selectSource = function(sources){
  for (var i=0,j=this.options_['techOrder'];i<j.length;i++) {
    var techName = vjs.capitalize(j[i]),
        tech = window['videojs'][techName];
    if (!tech) {
      vjs.log.error('The "' + techName + '" tech is undefined. Skipped browser support
check for that tech.');
      continue;
    }
    if (tech.isSupported()) {
      for (var a=0,b=sources;a<b.length;a++) {
        var source = b[a];
        if (tech['canPlaySource'](source)) {
          return { source: source, tech: techName };
        }
      }
    }
  }
  return false;
};
vjs.Player.prototype.src = function(source){
  if (source === undefined) {
    return this.techGet('src');
  }
  if (source instanceof Array) {
    var sourceTech = this.selectSource(source),
```

```
        techName;
    if (sourceTech) {
        source = sourceTech.source;
        techName = sourceTech.tech;
      if (techName == this.techName) {
        this.src(source); // Passing the source object
      } else {
        this.loadTech(techName, source);
      }
    } else {
      this.error({ code: 4, message: this.options()['notSupportedMessage'] });
      this.triggerReady(); // we could not find an appropriate tech, but let's still
notify the delegate that this is it
    }
  } else if (source instanceof Object) {
    if (window['videojs'][this.techName]['canPlaySource'](source)) {
      this.src(source.src);
    } else {
      this.src([source]);
    }
  } else {
    this.cache_.src = source;

    if (!this.isReady_) {
      this.ready(function(){
        this.src(source);
      });
    } else {
      this.techCall('src', source);
      if (this.options_['preload'] == 'auto') {
        this.load();
      }
      if (this.options_['autoplay']) {
        this.play();
      }
    }
  }
  return this;
};
vjs.Player.prototype.load = function(){
  this.techCall('load');
  return this;
};
vjs.Player.prototype.currentSrc = function(){
  return this.techGet('currentSrc') || this.cache_.src || '';
};
vjs.Player.prototype.preload = function(value){
  if (value !== undefined) {
    this.techCall('setPreload', value);
    this.options_['preload'] = value;
    return this;
  }
  return this.techGet('preload');
};
vjs.Player.prototype.autoplay = function(value){
  if (value !== undefined) {
    this.techCall('setAutoplay', value);
    this.options_['autoplay'] = value;
    return this;
  }
  return this.techGet('autoplay', value);
};
vjs.Player.prototype.loop = function(value){
  if (value !== undefined) {
    this.techCall('setLoop', value);
    this.options_['loop'] = value;
    return this;
  }
  return this.techGet('loop');
};
vjs.Player.prototype.poster_;
vjs.Player.prototype.poster = function(src){
  if (src === undefined) {
    return this.poster_;
```

```
  }
  this.poster_ = src;
  this.techCall('setPoster', src);
  this.trigger('posterchange');
};
vjs.Player.prototype.controls_;
vjs.Player.prototype.controls = function(bool){
  if (bool !== undefined) {
    bool = !!bool; // force boolean
    if (this.controls_ !== bool) {
      this.controls_ = bool;
      if (bool) {
        this.removeClass('vjs-controls-disabled');
        this.addClass('vjs-controls-enabled');
        this.trigger('controlsenabled');
      } else {
        this.removeClass('vjs-controls-enabled');
        this.addClass('vjs-controls-disabled');
        this.trigger('controlsdisabled');
      }
    }
    return this;
  }
  return this.controls_;
};
vjs.Player.prototype.usingNativeControls_;
vjs.Player.prototype.usingNativeControls = function(bool){
  if (bool !== undefined) {
    bool = !!bool; // force boolean
    if (this.usingNativeControls_ !== bool) {
      this.usingNativeControls_ = bool;
      if (bool) {
        this.addClass('vjs-using-native-controls');
        this.trigger('usingnativecontrols');
      } else {
        this.removeClass('vjs-using-native-controls');
        this.trigger('usingcustomcontrols');
      }
    }
    return this;
  }
  return this.usingNativeControls_;
};
vjs.Player.prototype.error_ = null;
vjs.Player.prototype.error = function(err){
  if (err === undefined) {
    return this.error_;
  }
  if (err === null) {
    this.error_ = err;
    this.removeClass('vjs-error');
    return this;
  }
  if (err instanceof vjs.MediaError) {
    this.error_ = err;
  } else {
    this.error_ = new vjs.MediaError(err);
  }
  this.trigger('error');
  this.addClass('vjs-error');
  vjs.log.error('(CODE:'+this.error_.code+'
'+vjs.MediaError.errorTypes[this.error_.code]+')', this.error_.message, this.error_);
  return this;
};
vjs.Player.prototype.ended = function(){ return this.techGet('ended'); };
vjs.Player.prototype.seeking = function(){ return this.techGet('seeking'); };
vjs.Player.prototype.userActivity_ = true;
vjs.Player.prototype.reportUserActivity = function(event){
  this.userActivity_ = true;
};
vjs.Player.prototype.userActive_ = true;
vjs.Player.prototype.userActive = function(bool){
  if (bool !== undefined) {
    bool = !!bool;
```

```
    if (bool !== this.userActive_) {
      this.userActive_ = bool;
      if (bool) {
        this.userActivity_ = true;
        this.removeClass('vjs-user-inactive');
        this.addClass('vjs-user-active');
        this.trigger('useractive');
      } else {
        this.userActivity_ = false;
        if(this.tech) {
          this.tech.one('mousemove', function(e){
            e.stopPropagation();
            e.preventDefault();
          });
        }
        this.removeClass('vjs-user-active');
        this.addClass('vjs-user-inactive');
        this.trigger('userinactive');
      }
    }
    return this;
  }
  return this.userActive_;
};
vjs.Player.prototype.listenForUserActivity = function(){
  var onActivity, onMouseMove, onMouseDown, mouseInProgress, onMouseUp,
      activityCheck, inactivityTimeout, lastMoveX, lastMoveY;
  onActivity = vjs.bind(this, this.reportUserActivity);
  onMouseMove = function(e) {
    if(e.screenX != lastMoveX || e.screenY != lastMoveY) {
      lastMoveX = e.screenX;
      lastMoveY = e.screenY;
      onActivity();
    }
  };
  onMouseDown = function() {
    onActivity();
    clearInterval(mouseInProgress);
    mouseInProgress = setInterval(onActivity, 250);
  };
  onMouseUp = function(event) {
    onActivity();
    clearInterval(mouseInProgress);
  };
  this.on('mousedown', onMouseDown);
  this.on('mousemove', onMouseMove);
  this.on('mouseup', onMouseUp);
  this.on('keydown', onActivity);
  this.on('keyup', onActivity);
  activityCheck = setInterval(vjs.bind(this, function() {
    if (this.userActivity_) {
      this.userActivity_ = false;
      this.userActive(true);
      clearTimeout(inactivityTimeout);
      inactivityTimeout = setTimeout(vjs.bind(this, function() {
        if (!this.userActivity_) {
          this.userActive(false);
        }
      }), 2000);
    }
  }), 250);
  this.on('dispose', function(){
    clearInterval(activityCheck);
    clearTimeout(inactivityTimeout);
  });
};
vjs.Player.prototype.playbackRate = function(rate) {
  if (rate !== undefined) {
    this.techCall('setPlaybackRate', rate);
    return this;
  }
  if (this.tech && this.tech.features && this.tech.features['playbackRate']) {
    return this.techGet('playbackRate');
  } else {
```

```
    return 1.0;
  }
};
```

js/tracks.js

- **487 lines of code**

```
vjs.Player.prototype.textTracks_;
vjs.Player.prototype.textTracks = function(){
  this.textTracks_ = this.textTracks_ || [];
  return this.textTracks_;
};
vjs.Player.prototype.addTextTrack = function(kind, label, language, options){
  var tracks = this.textTracks_ = this.textTracks_ || [];
  options = options || {};
  options['kind'] = kind;
  options['label'] = label;
  options['language'] = language;
  var Kind = vjs.capitalize(kind || 'subtitles');
  var track = new window['videojs'][Kind + 'Track'](this, options);
  tracks.push(track);
  if (track.dflt()) {
    this.ready(function(){
      setTimeout(function(){
        track.show();
      }, 0);
    });
  }
  return track;
};
vjs.Player.prototype.addTextTracks = function(trackList){
  var trackObj;
  for (var i = 0; i < trackList.length; i++) {
    trackObj = trackList[i];
    this.addTextTrack(trackObj['kind'], trackObj['label'], trackObj['language'],
trackObj);
  }
  return this;
};
vjs.Player.prototype.showTextTrack = function(id, disableSameKind){
  var tracks = this.textTracks_,
      i = 0,
      j = tracks.length, track, showTrack, kind;
  for (;i<j;i++) {
    track = tracks[i];
    if (track.id() === id) {
      track.show();
      showTrack = track;
    } else if (disableSameKind && track.kind() == disableSameKind && track.mode() > 0)
{
      track.disable();
    }
  }
  kind = (showTrack) ? showTrack.kind() : ((disableSameKind) ? disableSameKind :
false);
  if (kind) {
    this.trigger(kind+'trackchange');
  }
  return this;
};
vjs.TextTrack = vjs.Component.extend({
  init: function(player, options){
    vjs.Component.call(this, player, options);
    this.id_ = options['id'] || ('vjs_' + options['kind'] + '_' + options['language']
+ '_' + vjs.guid++);
    this.src_ = options['src'];
    this.dflt_ = options['default'] || options['dflt'];
    this.title_ = options['title'];
    this.language_ = options['srclang'];
    this.label_ = options['label'];
    this.cues_ = [];
    this.activeCues_ = [];
```

```
    this.readyState_ = 0;
    this.mode_ = 0;
    this.player_.on('fullscreenchange', vjs.bind(this, this.adjustFontSize));
  }
});
vjs.TextTrack.prototype.kind_;
vjs.TextTrack.prototype.kind = function(){
  return this.kind_;
};
vjs.TextTrack.prototype.src_;
vjs.TextTrack.prototype.src = function(){
  return this.src_;
};
vjs.TextTrack.prototype.dflt_;
vjs.TextTrack.prototype.dflt = function(){
  return this.dflt_;
};
vjs.TextTrack.prototype.title_;
vjs.TextTrack.prototype.title = function(){
  return this.title_;
};
vjs.TextTrack.prototype.language_;
vjs.TextTrack.prototype.language = function(){
  return this.language_;
};
vjs.TextTrack.prototype.label_;
vjs.TextTrack.prototype.label = function(){
  return this.label_;
};
vjs.TextTrack.prototype.cues_;
vjs.TextTrack.prototype.cues = function(){
  return this.cues_;
};
vjs.TextTrack.prototype.activeCues_;
vjs.TextTrack.prototype.activeCues = function(){
  return this.activeCues_;
};
vjs.TextTrack.prototype.readyState_;
vjs.TextTrack.prototype.readyState = function(){
  return this.readyState_;
};
vjs.TextTrack.prototype.mode_;
vjs.TextTrack.prototype.mode = function(){
  return this.mode_;
};
vjs.TextTrack.prototype.adjustFontSize = function(){
    if (this.player_.isFullScreen()) {
        this.el_.style.fontSize = screen.width / this.player_.width() * 1.4 * 100 +
'%';
    } else {
        this.el_.style.fontSize = '';
    }
};
vjs.TextTrack.prototype.createEl = function(){
  return vjs.Component.prototype.createEl.call(this, 'div', {
    className: 'vjs-' + this.kind_ + ' vjs-text-track'
  });
};
vjs.TextTrack.prototype.show = function(){
  this.activate();
  this.mode_ = 2;
  vjs.Component.prototype.show.call(this);
};
vjs.TextTrack.prototype.hide = function(){
  this.activate();
  this.mode_ = 1;
  vjs.Component.prototype.hide.call(this);
};
vjs.TextTrack.prototype.disable = function(){
  if (this.mode_ == 2) { this.hide(); }
  this.deactivate();
  this.mode_ = 0;
};
vjs.TextTrack.prototype.activate = function(){
```

```
  if (this.readyState_ === 0) { this.load(); }
  if (this.mode_ === 0) {
    this.player_.on('timeupdate', vjs.bind(this, this.update, this.id_));
    this.player_.on('ended', vjs.bind(this, this.reset, this.id_));
    if (this.kind_ === 'captions' || this.kind_ === 'subtitles') {
      this.player_.getChild('textTrackDisplay').addChild(this);
    }
  }
};
vjs.TextTrack.prototype.deactivate = function(){
  this.player_.off('timeupdate', vjs.bind(this, this.update, this.id_));
  this.player_.off('ended', vjs.bind(this, this.reset, this.id_));
  this.reset();
  this.player_.getChild('textTrackDisplay').removeChild(this);
};
vjs.TextTrack.prototype.load = function(){
  if (this.readyState_ === 0) {
    this.readyState_ = 1;
    vjs.get(this.src_, vjs.bind(this, this.parseCues), vjs.bind(this, this.onError));
  }
};
vjs.TextTrack.prototype.onError = function(err){
  this.error = err;
  this.readyState_ = 3;
  this.trigger('error');
};
vjs.TextTrack.prototype.parseCues = function(srcContent) {
  var cue, time, text,
      lines = srcContent.split('\n'),
      line = '', id;
  for (var i=1, j=lines.length; i<j; i++) {
    line = vjs.trim(lines[i]); // Trim whitespace and linebreaks
    if (line) { // Loop until a line with content
      if (line.indexOf('-->') == -1) {
        id = line;
        line = vjs.trim(lines[++i]);
      } else {
        id = this.cues_.length;
      }
      cue = {
        id: id, // Cue Number
        index: this.cues_.length // Position in Array
      };
      time = line.split(' --> ');
      cue.startTime = this.parseCueTime(time[0]);
      cue.endTime = this.parseCueTime(time[1]);
      text = [];
      while (lines[++i] && (line = vjs.trim(lines[i]))) {
        text.push(line);
      }
      cue.text = text.join('<br/>');
      this.cues_.push(cue);
    }
  }
  this.readyState_ = 2;
  this.trigger('loaded');
};
vjs.TextTrack.prototype.parseCueTime = function(timeText) {
  var parts = timeText.split(':'),
      time = 0,
      hours, minutes, other, seconds, ms;
  if (parts.length == 3) {
    hours = parts[0];
    minutes = parts[1];
    other = parts[2];
  } else {
    hours = 0;
    minutes = parts[0];
    other = parts[1];
  }
  other = other.split(/\s+/);
  seconds = other.splice(0,1)[0];
  seconds = seconds.split(/\.|,/);
  ms = parseFloat(seconds[1]);
```

```
  seconds = seconds[0];
  time += parseFloat(hours) * 3600;
  time += parseFloat(minutes) * 60;
  time += parseFloat(seconds);
  if (ms) { time += ms/1000; }
  return time;
};
vjs.TextTrack.prototype.update = function(){
  if (this.cues_.length > 0) {
    var offset = this.player_.options()['trackTimeOffset'] || 0;
    var time = this.player_.currentTime() + offset;
    if (this.prevChange === undefined || time < this.prevChange || this.nextChange <=
time) {
      var cues = this.cues_,
          newNextChange = this.player_.duration(), // Start at beginning of the
timeline
          newPrevChange = 0, // Start at end
          reverse = false, // Set the direction of the loop through the cues.
Optimized the cue check.
          newCues = [], // Store new active cues.
          firstActiveIndex, lastActiveIndex,
          cue, i; // Loop vars
      if (time >= this.nextChange || this.nextChange === undefined) { // NextChange
should happen
        i = (this.firstActiveIndex !== undefined) ? this.firstActiveIndex : 0;
      } else {
        reverse = true;
        i = (this.lastActiveIndex !== undefined) ? this.lastActiveIndex : cues.length
- 1;
      }
      while (true) { // Loop until broken
        cue = cues[i];
        if (cue.endTime <= time) {
          newPrevChange = Math.max(newPrevChange, cue.endTime);
          if (cue.active) {
            cue.active = false;
          }
        } else if (time < cue.startTime) {
          newNextChange = Math.min(newNextChange, cue.startTime);
          if (cue.active) {
            cue.active = false;
          }
          if (!reverse) { break; }
        } else {
          if (reverse) {
            newCues.splice(0,0,cue);
            if (lastActiveIndex === undefined) { lastActiveIndex = i; }
            firstActiveIndex = i;
          } else {
            newCues.push(cue);
            if (firstActiveIndex === undefined) { firstActiveIndex = i; }
            lastActiveIndex = i;
          }
          newNextChange = Math.min(newNextChange, cue.endTime);
          newPrevChange = Math.max(newPrevChange, cue.startTime);
          cue.active = true;
        }
        if (reverse) {
          if (i === 0) { break; } else { i--; }
        } else {
          if (i === cues.length - 1) { break; } else { i++; }
        }
      }
      this.activeCues_ = newCues;
      this.nextChange = newNextChange;
      this.prevChange = newPrevChange;
      this.firstActiveIndex = firstActiveIndex;
      this.lastActiveIndex = lastActiveIndex;
      this.updateDisplay();
      this.trigger('cuechange');
    }
  }
};
vjs.TextTrack.prototype.updateDisplay = function(){
  var cues = this.activeCues_,
```

```
      html = '',
      i=0,j=cues.length;
  for (;i<j;i++) {
    html += '<span class="vjs-tt-cue">'+cues[i].text+'</span>';
  }
  this.el_.innerHTML = html;
};
vjs.TextTrack.prototype.reset = function(){
  this.nextChange = 0;
  this.prevChange = this.player_.duration();
  this.firstActiveIndex = 0;
  this.lastActiveIndex = 0;
};
vjs.CaptionsTrack = vjs.TextTrack.extend();
vjs.CaptionsTrack.prototype.kind_ = 'captions';
vjs.SubtitlesTrack = vjs.TextTrack.extend();
vjs.SubtitlesTrack.prototype.kind_ = 'subtitles';
vjs.ChaptersTrack = vjs.TextTrack.extend();
vjs.ChaptersTrack.prototype.kind_ = 'chapters';
vjs.TextTrackDisplay = vjs.Component.extend({
  init: function(player, options, ready){
    vjs.Component.call(this, player, options, ready);
    if (player.options_['tracks'] && player.options_['tracks'].length > 0) {
      this.player_.addTextTracks(player.options_['tracks']);
    }
  }
});
vjs.TextTrackDisplay.prototype.createEl = function(){
  return vjs.Component.prototype.createEl.call(this, 'div', {
    className: 'vjs-text-track-display'
  });
};
vjs.TextTrackMenuItem = vjs.MenuItem.extend({
  init: function(player, options){
    var track = this.track = options['track'];
    options['label'] = track.label();
    options['selected'] = track.dflt();
    vjs.MenuItem.call(this, player, options);
    this.player_.on(track.kind() + 'trackchange', vjs.bind(this, this.update));
  }
});
vjs.TextTrackMenuItem.prototype.onClick = function(){
  vjs.MenuItem.prototype.onClick.call(this);
  this.player_.showTextTrack(this.track.id_, this.track.kind());
};
vjs.TextTrackMenuItem.prototype.update = function(){
  this.selected(this.track.mode() == 2);
};
vjs.OffTextTrackMenuItem = vjs.TextTrackMenuItem.extend({
  init: function(player, options){
    options['track'] = {
      kind: function() { return options['kind']; },
      player: player,
      label: function(){ return options['kind'] + ' off'; },
      dflt: function(){ return false; },
      mode: function(){ return false; }
    };
    vjs.TextTrackMenuItem.call(this, player, options);
    this.selected(true);
  }
});
vjs.OffTextTrackMenuItem.prototype.onClick = function(){
  vjs.TextTrackMenuItem.prototype.onClick.call(this);
  this.player_.showTextTrack(this.track.id_, this.track.kind());
};
vjs.OffTextTrackMenuItem.prototype.update = function(){
  var tracks = this.player_.textTracks(),
      i=0, j=tracks.length, track,
      off = true;
  for (;i<j;i++) {
    track = tracks[i];
    if (track.kind() == this.track.kind() && track.mode() == 2) {
      off = false;
    }
```

```
  }
  this.selected(off);
};
vjs.TextTrackButton = vjs.MenuButton.extend({
  init: function(player, options){
    vjs.MenuButton.call(this, player, options);
    if (this.items.length <= 1) {
      this.hide();
    }
  }
});
vjs.TextTrackButton.prototype.createItems = function(){
  var items = [], track;
  items.push(new vjs.OffTextTrackMenuItem(this.player_, { 'kind': this.kind_ }));
  for (var i = 0; i < this.player_.textTracks().length; i++) {
    track = this.player_.textTracks()[i];
    if (track.kind() === this.kind_) {
      items.push(new vjs.TextTrackMenuItem(this.player_, {
        'track': track
      }));
    }
  }
  return items;
};
vjs.CaptionsButton = vjs.TextTrackButton.extend({
  init: function(player, options, ready){
    vjs.TextTrackButton.call(this, player, options, ready);
    this.el_.setAttribute('aria-label','Captions Menu');
  }
});
vjs.CaptionsButton.prototype.kind_ = 'captions';
vjs.CaptionsButton.prototype.buttonText = 'Captions';
vjs.CaptionsButton.prototype.className = 'vjs-captions-button';
vjs.SubtitlesButton = vjs.TextTrackButton.extend({
  init: function(player, options, ready){
    vjs.TextTrackButton.call(this, player, options, ready);
    this.el_.setAttribute('aria-label','Subtitles Menu');
  }
});
vjs.SubtitlesButton.prototype.kind_ = 'subtitles';
vjs.SubtitlesButton.prototype.buttonText = 'Subtitles';
vjs.SubtitlesButton.prototype.className = 'vjs-subtitles-button';
vjs.ChaptersButton = vjs.TextTrackButton.extend({
  init: function(player, options, ready){
    vjs.TextTrackButton.call(this, player, options, ready);
    this.el_.setAttribute('aria-label','Chapters Menu');
  }
});
vjs.ChaptersButton.prototype.kind_ = 'chapters';
vjs.ChaptersButton.prototype.buttonText = 'Chapters';
vjs.ChaptersButton.prototype.className = 'vjs-chapters-button';
vjs.ChaptersButton.prototype.createItems = function(){
  var items = [], track;
  for (var i = 0; i < this.player_.textTracks().length; i++) {
    track = this.player_.textTracks()[i];
    if (track.kind() === this.kind_) {
      items.push(new vjs.TextTrackMenuItem(this.player_, {
        'track': track
      }));
    }
  }
  return items;
};
vjs.ChaptersButton.prototype.createMenu = function(){
  var tracks = this.player_.textTracks(),
      i = 0,
      j = tracks.length,
      track, chaptersTrack,
      items = this.items = [];
  for (;i<j;i++) {
    track = tracks[i];
    if (track.kind() == this.kind_ && track.dflt()) {
      if (track.readyState() < 2) {
        this.chaptersTrack = track;
```

```
        track.on('loaded', vjs.bind(this, this.createMenu));
        return;
      } else {
        chaptersTrack = track;
        break;
      }
    }
  }
  var menu = this.menu = new vjs.Menu(this.player_);
  menu.contentEl().appendChild(vjs.createEl('li', {
    className: 'vjs-menu-title',
    innerHTML: vjs.capitalize(this.kind_),
    tabindex: -1
  }));
  if (chaptersTrack) {
    var cues = chaptersTrack.cues_, cue, mi;
    i = 0;
    j = cues.length;
    for (;i<j;i++) {
      cue = cues[i];
      mi = new vjs.ChaptersTrackMenuItem(this.player_, {
        'track': chaptersTrack,
        'cue': cue
      });
      items.push(mi);
      menu.addChild(mi);
    }
  }
  if (this.items.length > 0) {
    this.show();
  }
  return menu;
};
vjs.ChaptersTrackMenuItem = vjs.MenuItem.extend({
  init: function(player, options){
    var track = this.track = options['track'],
        cue = this.cue = options['cue'],
        currentTime = player.currentTime();
    options['label'] = cue.text;
    options['selected'] = (cue.startTime <= currentTime && currentTime < cue.endTime);
    vjs.MenuItem.call(this, player, options);
    track.on('cuechange', vjs.bind(this, this.update));
  }
});
vjs.ChaptersTrackMenuItem.prototype.onClick = function(){
  vjs.MenuItem.prototype.onClick.call(this);
  this.player_.currentTime(this.cue.startTime);
  this.update(this.cue.startTime);
};
vjs.ChaptersTrackMenuItem.prototype.update = function(){
  var cue = this.cue,
      currentTime = this.player_.currentTime();
  this.selected(cue.startTime <= currentTime && currentTime < cue.endTime);
};
vjs.obj.merge(vjs.ControlBar.prototype.options_['children'], {
  'subtitlesButton': {},
  'captionsButton': {},
  'chaptersButton': {}
});
```

js/control-bar/volume-menu-button.js

- **38 lines of code**

```
vjs.VolumeMenuButton = vjs.MenuButton.extend({
  init: function(player, options){
    vjs.MenuButton.call(this, player, options);
    player.on('volumechange', vjs.bind(this, this.update));
    if (player.tech && player.tech.features && player.tech.features.volumeControl ===
false) {
      this.addClass('vjs-hidden');
    }
    player.on('loadstart', vjs.bind(this, function(){
      if (player.tech.features && player.tech.features.volumeControl === false) {
```

```
        this.addClass('vjs-hidden');
      } else {
        this.removeClass('vjs-hidden');
      }
    }));
    this.addClass('vjs-menu-button');
  }
});
vjs.VolumeMenuButton.prototype.createMenu = function(){
  var menu = new vjs.Menu(this.player_, {
    contentElType: 'div'
  });
  var vc = new vjs.VolumeBar(this.player_, vjs.obj.merge({vertical: true},
this.options_.volumeBar));
  menu.addChild(vc);
  return menu;
};
vjs.VolumeMenuButton.prototype.onClick = function(){
  vjs.MuteToggle.prototype.onClick.call(this);
  vjs.MenuButton.prototype.onClick.call(this);
};
vjs.VolumeMenuButton.prototype.createEl = function(){
  return vjs.Button.prototype.createEl.call(this, 'div', {
    className: 'vjs-volume-menu-button vjs-menu-button vjs-control',
    innerHTML: '<div><span class="vjs-control-text">Mute</span></div>'
  });
};
vjs.VolumeMenuButton.prototype.update = vjs.MuteToggle.prototype.update;
```

js/media/flash.js

- 322 lines of code

```
vjs.Flash = vjs.MediaTechController.extend({
  init: function(player, options, ready){
    vjs.MediaTechController.call(this, player, options, ready);
    var source = options['source'],
        parentEl = options['parentEl'],
        placeHolder = this.el_ = vjs.createEl('div', { id: player.id() + '_temp_flash'
}),
        objId = player.id()+'_flash_api',
        playerOptions = player.options_,
        flashVars = vjs.obj.merge({
          'readyFunction': 'videojs.Flash.onReady',
          'eventProxyFunction': 'videojs.Flash.onEvent',
          'errorEventProxyFunction': 'videojs.Flash.onError',
          'autoplay': playerOptions.autoplay,
          'preload': playerOptions.preload,
          'loop': playerOptions.loop,
          'muted': playerOptions.muted
        }, options['flashVars']),
        params = vjs.obj.merge({
          'wmode': 'opaque', // Opaque is needed to overlay controls, but can affect
playback performance
          'bgcolor': '#000000' // Using bgcolor prevents a white flash when the object
is loading
        }, options['params']),
        attributes = vjs.obj.merge({
          'id': objId,
          'name': objId, // Both ID and Name needed or swf to identifty itself
          'class': 'vjs-tech'
        }, options['attributes']),
        lastSeekTarget
    ;
    if (source) {
      if (source.type && vjs.Flash.isStreamingType(source.type)) {
        var parts = vjs.Flash.streamToParts(source.src);
        flashVars['rtmpConnection'] = encodeURIComponent(parts.connection);
        flashVars['rtmpStream'] = encodeURIComponent(parts.stream);
      }
      else {
        flashVars['src'] = encodeURIComponent(vjs.getAbsoluteURL(source.src));
      }
    }
```

```
    this['setCurrentTime'] = function(time){
      lastSeekTarget = time;
      this.el_.vjs_setProperty('currentTime', time);
    };
    this['currentTime'] = function(time){
      if (this.seeking()) {
        return lastSeekTarget;
      }
      return this.el_.vjs_getProperty('currentTime');
    };
    vjs.insertFirst(placeHolder, parentEl);
    if (options['startTime']) {
      this.ready(function(){
        this.load();
        this.play();
        this.currentTime(options['startTime']);
      });
    }
    if (vjs.IS_FIREFOX) {
      this.ready(function(){
        vjs.on(this.el(), 'mousemove', vjs.bind(this, function(){
          // since it's a custom event, don't bubble higher than the player
          this.player().trigger({ 'type':'mousemove', 'bubbles': false });
        }));
      });
    }
    if (options['iFrameMode'] === true && !vjs.IS_FIREFOX) {
      var iFrm = vjs.createEl('iframe', {
        'id': objId + '_iframe',
        'name': objId + '_iframe',
        'className': 'vjs-tech',
        'scrolling': 'no',
        'marginWidth': 0,
        'marginHeight': 0,
        'frameBorder': 0
      });
      flashVars['readyFunction'] = 'ready';
      flashVars['eventProxyFunction'] = 'events';
      flashVars['errorEventProxyFunction'] = 'errors';
      vjs.on(iFrm, 'load', vjs.bind(this, function(){
        var iDoc,
            iWin = iFrm.contentWindow;
        iDoc = iFrm.contentDocument ? iFrm.contentDocument :
iFrm.contentWindow.document;
        iDoc.write(vjs.Flash.getEmbedCode(options['swf'], flashVars, params,
attributes));
        iWin['player'] = this.player_;
        iWin['ready'] = vjs.bind(this.player_, function(currSwf){
          var el = iDoc.getElementById(currSwf),
              player = this,
              tech = player.tech;
          tech.el_ = el;
          vjs.Flash.checkReady(tech);
        });
        iWin['events'] = vjs.bind(this.player_, function(swfID, eventName){
          var player = this;
          if (player && player.techName === 'flash') {
            player.trigger(eventName);
          }
        });
        iWin['errors'] = vjs.bind(this.player_, function(swfID, eventName){
          vjs.log('Flash Error', eventName);
        });
      }));
      placeHolder.parentNode.replaceChild(iFrm, placeHolder);
    } else {
      vjs.Flash.embed(options['swf'], placeHolder, flashVars, params, attributes);
    }
  }
});
vjs.Flash.prototype.dispose = function(){
  vjs.MediaTechController.prototype.dispose.call(this);
};
vjs.Flash.prototype.play = function(){
```

```
  this.el_.vjs_play();
};
vjs.Flash.prototype.pause = function(){
  this.el_.vjs_pause();
};
vjs.Flash.prototype.src = function(src){
  if (src === undefined) {
    return this.currentSrc();
  }
  if (vjs.Flash.isStreamingSrc(src)) {
    src = vjs.Flash.streamToParts(src);
    this.setRtmpConnection(src.connection);
    this.setRtmpStream(src.stream);
  } else {
    src = vjs.getAbsoluteURL(src);
    this.el_.vjs_src(src);
  }
  if (this.player_.autoplay()) {
    var tech = this;
    setTimeout(function(){ tech.play(); }, 0);
  }
};
vjs.Flash.prototype.currentSrc = function(){
  var src = this.el_.vjs_getProperty('currentSrc');
  if (src == null) {
    var connection = this['rtmpConnection'](),
        stream = this['rtmpStream']();
    if (connection && stream) {
      src = vjs.Flash.streamFromParts(connection, stream);
    }
  }
  return src;
};
vjs.Flash.prototype.load = function(){
  this.el_.vjs_load();
};
vjs.Flash.prototype.poster = function(){
  this.el_.vjs_getProperty('poster');
};
vjs.Flash.prototype.setPoster = function(){
};
vjs.Flash.prototype.buffered = function(){
  return vjs.createTimeRange(0, this.el_.vjs_getProperty('buffered'));
};
vjs.Flash.prototype.supportsFullScreen = function(){
  return false; // Flash does not allow fullscreen through javascript
};
vjs.Flash.prototype.enterFullScreen = function(){
  return false;
};
var api = vjs.Flash.prototype,
    readWrite =
'rtmpConnection,rtmpStream,preload,defaultPlaybackRate,playbackRate,autoplay,loop,medi
aGroup,controller,controls,volume,muted,defaultMuted'.split(','),
    readOnly =
'error,networkState,readyState,seeking,initialTime,duration,startOffsetTime,paused,pla
yed,seekable,ended,videoTracks,audioTracks,videoWidth,videoHeight,textTracks'.split(',
');
var createSetter = function(attr){
  var attrUpper = attr.charAt(0).toUpperCase() + attr.slice(1);
  api['set'+attrUpper] = function(val){ return this.el_.vjs_setProperty(attr, val); };
};
var createGetter = function(attr){
  api[attr] = function(){ return this.el_.vjs_getProperty(attr); };
};
(function(){
  var i;
  for (i = 0; i < readWrite.length; i++) {
    createGetter(readWrite[i]);
    createSetter(readWrite[i]);
  }
  for (i = 0; i < readOnly.length; i++) {
    createGetter(readOnly[i]);
  }
})();
```

```
vjs.Flash.isSupported = function(){
  return vjs.Flash.version()[0] >= 10;
};
vjs.Flash.canPlaySource = function(srcObj){
  var type;
  if (!srcObj.type) {
    return '';
  }
  type = srcObj.type.replace(/;.*/,'').toLowerCase();
  if (type in vjs.Flash.formats || type in vjs.Flash.streamingFormats) {
    return 'maybe';
  }
};
vjs.Flash.formats = {
  'video/flv': 'FLV',
  'video/x-flv': 'FLV',
  'video/mp4': 'MP4',
  'video/m4v': 'MP4'
};
vjs.Flash.streamingFormats = {
  'rtmp/mp4': 'MP4',
  'rtmp/flv': 'FLV'
};
vjs.Flash['onReady'] = function(currSwf){
  var el = vjs.el(currSwf);
  var player = el['player'] || el.parentNode['player'],
      tech = player.tech;
  el['player'] = player;
  tech.el_ = el;
  vjs.Flash.checkReady(tech);
};
vjs.Flash.checkReady = function(tech){
  if (tech.el().vjs_getProperty) {
    tech.triggerReady();
  } else {
    setTimeout(function(){
      vjs.Flash.checkReady(tech);
    }, 50);
  }
};
vjs.Flash['onEvent'] = function(swfID, eventName){
  var player = vjs.el(swfID)['player'];
  player.trigger(eventName);
};
vjs.Flash['onError'] = function(swfID, err){
  var player = vjs.el(swfID)['player'];
  var msg = 'FLASH: '+err;
  if (err == 'srcnotfound') {
    player.error({ code: 4, message: msg });
  } else {
    player.error(msg);
  }
};
vjs.Flash.version = function(){
  var version = '0,0,0';
  try {
    version = new
window.ActiveXObject('ShockwaveFlash.ShockwaveFlash').GetVariable('$version').replace(
/\D+/g, ',').match(/^,?(.+),?$/)[1];
  } catch(e) {
    try {
      if (navigator.mimeTypes['application/x-shockwave-flash'].enabledPlugin){
        version = (navigator.plugins['Shockwave Flash 2.0'] ||
navigator.plugins['Shockwave Flash']).description.replace(/\D+/g,
',').match(/^,?(.+),?$/)[1];
      }
    } catch(err) {}
  }
  return version.split(',');
};
vjs.Flash.embed = function(swf, placeHolder, flashVars, params, attributes){
  var code = vjs.Flash.getEmbedCode(swf, flashVars, params, attributes),
    obj = vjs.createEl('div', { innerHTML: code }).childNodes[0],
      par = placeHolder.parentNode
```

```
  ;
  placeHolder.parentNode.replaceChild(obj, placeHolder);
  var newObj = par.childNodes[0];
  setTimeout(function(){
    newObj.style.display = 'block';
  }, 1000);
  return obj;
};
vjs.Flash.getEmbedCode = function(swf, flashVars, params, attributes){
  var objTag = '<object type="application/x-shockwave-flash"',
      flashVarsString = '',
      paramsString = '',
      attrsString = '';
  if (flashVars) {
    vjs.obj.each(flashVars, function(key, val){
      flashVarsString += (key + '=' + val + '&amp;');
    });
  }
  params = vjs.obj.merge({
    'movie': swf,
    'flashvars': flashVarsString,
    'allowScriptAccess': 'always', // Required to talk to swf
    'allowNetworking': 'all' // All should be default, but having security issues.
  }, params);
  vjs.obj.each(params, function(key, val){
    paramsString += '<param name="'+key+'" value="'+val+'" />';
  });
  attributes = vjs.obj.merge({
    'data': swf,
    'width': '100%',
    'height': '100%'
  }, attributes);
  vjs.obj.each(attributes, function(key, val){
    attrsString += (key + '="' + val + '" ');
  });
  return objTag + attrsString + '>' + paramsString + '</object>';
};
vjs.Flash.streamFromParts = function(connection, stream) {
  return connection + '&' + stream;
};
vjs.Flash.streamToParts = function(src) {
  var parts = {
    connection: '',
    stream: ''
  };
  if (! src) {
    return parts;
  }
  var connEnd = src.indexOf('&');
  var streamBegin;
  if (connEnd !== -1) {
    streamBegin = connEnd + 1;
  }
  else {
    connEnd = streamBegin = src.lastIndexOf('/') + 1;
    if (connEnd === 0) {
      connEnd = streamBegin = src.length;
    }
  }
  parts.connection = src.substring(0, connEnd);
  parts.stream = src.substring(streamBegin, src.length);
  return parts;
};
vjs.Flash.isStreamingType = function(srcType) {
  return srcType in vjs.Flash.streamingFormats;
};
vjs.Flash.RTMP_RE = /^rtmp[set]?:\/\//i;
vjs.Flash.isStreamingSrc = function(src) {
  return vjs.Flash.RTMP_RE.test(src);
};
```

js/core.js

- **58 lines of code**

```
document.createElement('video');
document.createElement('audio');
document.createElement('track');
var vjs = function(id, options, ready){
  var tag; // Element of ID
  if (typeof id === 'string') {
    if (id.indexOf('#') === 0) {
      id = id.slice(1);
    }
    if (vjs.players[id]) {
      return vjs.players[id];
    } else {
      tag = vjs.el(id);
    }
  } else {
    tag = id;
  }
  if (!tag || !tag.nodeName) { // re: nodeName, could be a box div also
    throw new TypeError('The element or ID supplied is not valid. (videojs)'); //
Returns
  }
  return tag['player'] || new vjs.Player(tag, options, ready);
};
var videojs = vjs;
window.videojs = window.vjs = vjs;
vjs.CDN_VERSION = 'GENERATED_CDN_VSN';
vjs.ACCESS_PROTOCOL = ('https:' == document.location.protocol ? 'https://' :
'http://');
vjs.options = {
  'techOrder': ['html5','flash'],
  'html5': {},
  'flash': {},
  'width': 300,
  'height': 150,
  'defaultVolume': 0.00, // The freakin seaguls are driving me crazy!
  'playbackRates': [],
  'children': {
    'mediaLoader': {},
    'posterImage': {},
    'textTrackDisplay': {},
    'loadingSpinner': {},
    'bigPlayButton': {},
    'controlBar': {},
    'errorDisplay': {}
  },
  'notSupportedMessage': 'No compatible source was found for this video.'
};
if (vjs.CDN_VERSION !== 'GENERATED'+'_CDN_VSN') {
  videojs.options['flash']['swf'] = vjs.ACCESS_PROTOCOL +
'vjs.zencdn.net/'+vjs.CDN_VERSION+'/video-js.swf';
}
vjs.players = {};
if (typeof define === 'function' && define['amd']) {
  define([], function(){ return videojs; });
} else if (typeof exports === 'object' && typeof module === 'object') {
  module['exports'] = videojs;
}
```