Get started with Video.js

This project is based off of the video streaming project I wrote about last year. Since that project was set to serve RTMP streams, to use Video.js, you'll need to make some adjustments to that Nginx configuration. HTTP Live Streaming (HLS) is a widely used protocol developed by Apple that will serve your stream better to a multitude of devices. HLS will take your stream, break it into chunks, and serve it via a specialized playlist. This allows for a more fault-tolerant stream that can play on more devices.

First, create a directory that will house the HLS stream and give Nginx permission to write to it:

```
mkdir /mnt/hls
chown www:www /mnt/hls
```

Next, fire up your text editor, open the Nginx.conf file, and add the following under the **application live** section:

```
        application live {
             live on;
             # Turn on HLS
             hls on;
             hls_path /mnt/hls/;
             hls_fragment 3;
             hls_playlist_length 60;
             # disable consuming the stream from nginx as rtmp
             deny play all;
}
```

Take note of the HLS fragment and playlist length settings. You may want to adjust them later, depending on your streaming needs, but this is a good baseline to start with. Next, we need to ensure that Nginx is able to listen for requests from our player and understand how to present it to the user. So, we'll want to add a new section at the bottom of our nginx.conf file.

```
server {
        listen 8080;

        location / {
            # Disable cache
            add_header 'Cache-Control' 'no-cache';

            # CORS setup
            add_header 'Access-Control-Allow-Origin' '*' always;
            add_header 'Access-Control-Expose-Headers' 'Content-Length';

            # allow CORS preflight requests
            if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Max-Age' 1728000;
                add_header 'Content-Type' 'text/plain charset=UTF-8';
                add_header 'Content-Length' 0;
                return 204;
            }

            types {
                application/dash+xml mpd;
                application/vnd.apple.mpegurl m3u8;
                video/mp2t ts;
            }

            root /mnt/;
        }
    }
```

Visit Video.js's Getting started page to download the latest release and check out the release notes. Also on that page, Video.js has a great introductory template you can use to create a very basic web player. I'll break down the important bits of that template and insert the pieces you need to get your new HTML player to use your stream.

The **head** links in the Video.js library from a content-delivery network (CDN). You can also opt to download and store Video.js locally on your web server if you want.

```
<head>
  <link href="https://vjs.zencdn.net/7.5.5/video-js.css" rel="stylesheet" />

  <!-- If you'd like to support IE8 (for Video.js versions prior to v7) -->
```

```
  <script src="https://vjs.zencdn.net/ie8/1.1.2/videojs-ie8.min.js"></script>
</head>
```

Now to the real meat of the player. The **body** section sets the parameters of how the video player will be displayed. Within the **video** element, you need to define the properties of your player. How big do you want it to be? Do you want it to have a poster (i.e., a thumbnail)? Does it need any special player controls? This example defines a simple 600x600 pixel player with an appropriate (to me) thumbnail featuring Beastie (the BSD Demon) and Tux (the Linux penguin).

```
<body>
  <video
    id="my-video"
    class="video-js"
    controls
    preload="auto"
    width="600"
    height="600"
    poster="BEASTIE-TUX.jpg"
    data-setup="{}"
  >
```

Now that you've set how you want your player to look, you need to tell it what to play. Video.js can handle a large number of different formats, including HLS streams.

```
    <source src="http://MY-WEB-SERVER:8080/hls/STREAM-KEY.m3u8" type="application/x-mpegURL'
    <p class="vjs-no-js">
      To view this video please enable JavaScript, and consider upgrading to a
      web browser that
      <a href="https://videojs.com/html5-video-support/" target="_blank"
        >supports HTML5 video</a
      >
    </p>
  </video>
```

Record your streams

Keeping a copy of your streams is super easy. Just add the following at the bottom of your **application live** section in the nginx.conf file:

```
# Enable stream recording
record all;
record_path /mnt/recordings/;
record_unique on;
```

Make sure that **record_path** exists and that Nginx has permissions to write to it:

```
chown -R www:www /mnt/recordings
```

Down the stream

That's it! You should now have a spiffy new HTML5-friendly live video player. There are lots of great resources out there on how to expand all your video-making adventures. If you have any questions or suggestions, feel free to reach out to me on Twitter or leave a comment below.