Working with Video

Sourcing video from a host

The easiest method for including video on a Gatsby site is to source an uploaded file from a site like YouTube, Vimeo, or Twitch. Using the source URL from one of those hosts, you can use Remark plugins or create a custom `<iframe>` solution to embed videos into your Gatsby site.

Embedding hosted videos in Markdown

There are numerous Gatsby plugins for working with hosted video in your Markdown posts and pages. We recommend checking out the gatsby-remark-embed-video plugin for sourcing from a variety of hosts like YouTube or Vimeo.

Writing custom components for hosted video

If you would like more control over how YouTube (or similar) videos are embedded into your Gatsby posts and pages, you can write a reusable custom `iframe` component and include it in a JSX template or in your content with MDX.

In this reusable sample component, you could include props for video data like URL or title, any necessary markup for styling purposes, and the common `iframe` embed code:

```
import React from "react"
const Video = ({ videoSrcURL, videoTitle, ...props }) => (
  <div className="video">
    <iframe
      src={videoSrcURL}
      title={videoTitle}
      allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture"
      frameBorder="0"
      webkitallowfullscreen="true"
      mozallowfullscreen="true"
      allowFullScreen
    />
  </div>
)
export default Video
```

You would then include this component in a template or page with a video source URL and title passed in as props. The data for video URLs and titles can be sourced in multiple ways, such as importing JSON or querying video data from Markdown with GraphQL. You can also hard-code video data for something fun, like a custom 404 page with an Easter egg YouTube video:

```
import React from "react"

import Layout from "../components/layout"
import SEO from "../components/seo"
import Video from "../components/video"

const NotFoundPage = () => (
  <Layout>
    <SEO title="404: Not found" />
    <section>
      <h1>NOT FOUND</h1>
      <p>You just hit a page that doesn't exist... the sadness.</p>
      <p>May I suggest a video instead?</p>
      <Video
        videoSrcURL="https://www.youtube.com/embed/dQw4w9WgXcQ"
        videoTitle="Official Music Video on YouTube"
      />
    </section>
  </Layout>
)

export default NotFoundPage
```

Querying video data from Markdown with GraphQL

If a Markdown page or post has a featured video, you might want to include a video URL and title in its frontmatter. This allows you to pass those values into your custom component:

```
---
path: "/blog/my-first-post"
date: "2019-03-27"
title: "My first blog post"
videoSourceURL: https://www.youtube.com/embed/dQw4w9WgXcQ
videoTitle: "Gatsby is Never Gonna Give You Up"
---
```

To include a video component in a template, you could start with something like this:

```
import React from "react"
import { graphql } from "gatsby"

import Video from "../components/video"

export default function VlogTemplate({
  data, // this prop will be injected by the GraphQL query below.
}) {
  const { markdownRemark } = data // data.markdownRemark holds your post data
  const { frontmatter, html } = markdownRemark
  return (
    <div className="blog-post-container">
      <div className="blog-post">
        <h1>{frontmatter.title}</h1>
        <h2>{frontmatter.date}</h2>
        <Video
          videoSrcURL={frontmatter.videoSrcURL}
          videoTitle={frontmatter.videoTitle}
        />
        <div
          className="blog-post-content"
          dangerouslySetInnerHTML={{ __html: html }}
        />
      </div>
    </div>
  )
}

export const pageQuery = graphql`
  query($path: String!) {
    markdownRemark(frontmatter: { path: { eq: $path } }) {
      html
      frontmatter {
        date(formatString: "MMMM DD, YYYY")
        path
        title
        videoSrcURL
        videoTitle
      }
    }
  }
`
```

Hosting your own HTML5 video files

It's super common to source video from YouTube, Twitch or Vimeo. But what if you want to host your own video and include it as HTML5 video?

To include your own video files that will work in multiple web browsers and platforms, you'll need to read up a bit on video extensions and codecs. We recommend MDN as a resource: Media formats for HTML audio and video. You may need video converter software to produce the necessary formats – such as `.webm` and `.mp4` – to support a range of devices and environments.

HTML5 provides the `<video>` media element for working with videos. Inside the `<video>` element, you can provide multiple `<source>` elements that serve as different file formats the video player can use, with each browser electing to use a format it supports.

If you have a video called `dog.mp4` in your site under `src/assets/dog.mp4`, you can include that video in your page with webpack like you would other assets. Then reference it in a `<source>` element, which is wrapped by a `<video>` element:

```
import React from "react"
import DogVideo from "../assets/dog.mp4"

export default function Home() {
  return (
    <video controls>
      <source src={DogVideo} type="video/mp4" /> // highlight-line
    </video>
  );
}
```

The `controls` attribute on the `<video>` will provide a default set of buttons overlaid on the video to play/pause, adjust volume, and go full screen. Other attributes like `muted` can set audio to silent, or `poster` can display an image when the video isn't playing. Common attributes that you'd want to apply to multiple videos could be extracted into a custom React video component. A full list of `<video>` attributes can be found on the MDN docs.

Supporting multiple browsers and formats

Adding more source tags for additional formats will allow the browser to find a source type that it supports, if there are no matching source types the video will fail to load. You can see what formats are supported in different browsers on MDN's docs about supported media formats.

```
import React from "react"
import DogMp4 from "../assets/dog.mp4"
import DogOgg from "../assets/dog.ogg" // highlight-line
```

```
export default function Home() {
  return (
    <video controls>
      <source src={DogMp4} type="video/mp4" />
      <source src={DogOgg} type="video/ogg" /> // highlight-line
    </video>
  );
}
```

Even though there are two `<source>` elements, only one video will be displayed, first `mp4` if it is supported, then `.ogg`.

Note: This requires importing a video in the format of the type specified, i.e. adding a `<source>` element with `type=video/ogg` would also need a file import with a format of `.ogg`. Alternatively, you can specify a URL to where a video is remotely hosted as the `src` instead of importing a local file.

See an example repository using `<video>` elements

Accessibility with custom video players

One advantage of integrating a custom component with your own hosted video is it can give you more control over the video player, including its accessibility. Elements of accessible video and audio include:

- captions: a text version of the audio, synchronized with the video
- transcript (or subtitles): a text version of the audio and visual content, like captions but also including descriptions of key visual elements in the video
- audio description: an audio version of visual information not conveyed in dialogue
- accessible controls: buttons to operate the video that can be operated without a mouse, are labeled, and work across environments and browsers

Though captions, transcripts, and audio descriptions primarily aim to assist those with greater difficulty seeing or hearing, they benefit many other users who prefer reading to listening. Captions can also help people watching videos when they can't have the sound turned on for whatever reason.

HTML5 provides support for these types of assistive content through the `<track>` element. The track element is nested under a `<video>` element as an empty tag. An example usage of the `<track>` element with a video looks like this:

```
import React from "react"
import DogMp4 from "../assets/dog.mp4"
import Captions from "file-loader!../assets/captions.vtt" // highlight-line

export default function Home() {
  return (
    <video controls>
      <source src={DogMp4} type="video/mp4" />
      // highlight-start
      <track kind="captions" srcLang="en" src={Captions} />
      // highlight-end
    </video>
  )
}
```

The kind attribute can be of a variety of different types including `captions`, `subtitles`, and `descriptions`, among others. The `srcLang` defines English as the language used in the captions in the example, and the captions file imported is used as the source. You can read about the specific attributes of a `<track>` on MDN.

Note: The filepath to import the captions in the above code snippet includes the `file-loader!` prefix, which helps webpack import the `.vtt` caption file.

Check out the accessible HTML5 video player from PayPal for an example compatible with Gatsby and React.