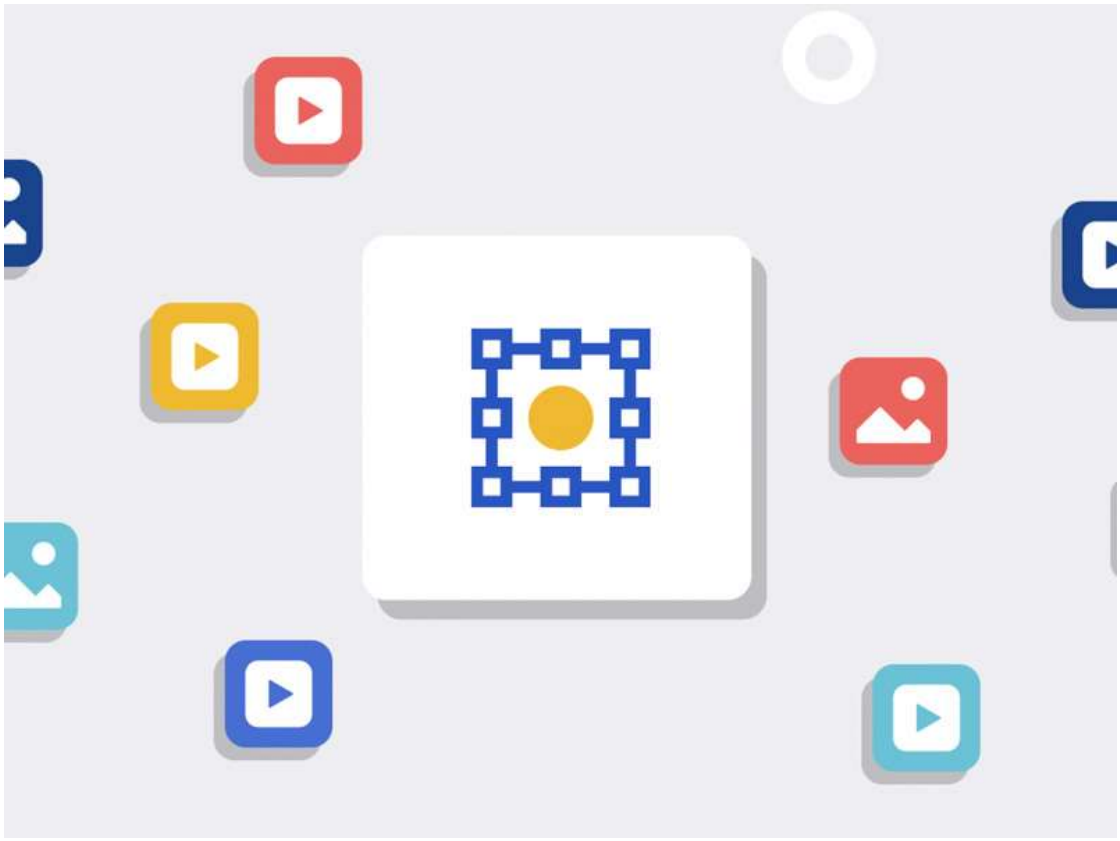
Ten Useful Image and Video Transformations

Learn how to transform images and videos for 10 popular use cases, such as image thumbnails and placeholders; as well as video transitions and previews.

In Articles on May 3, 2021 — **Sponsored**



Cloudinary is an end-to-end image- and video-management solution for websites and mobile apps, covering everything from uploads, storage, transformations, optimizations to delivery. Additionally, Cloudinary offers comprehensive APIs and administration capabilities that seamlessly integrate with web and mobile apps.

This tutorial steps you through 10 interesting and useful image and video transformations, which you perform on the client side (browser) through Cloudinary's JavaScript SDK. At the end of this tutorial, you'll see the transformations live and their real-world use cases, which make for significant time-savers for software development.

Integration With JavaScript

To start, install the necessary files with the NPM package manager:

```
npm install cloudinary-core --save
```

The optional –save parameter saves the dependency in your package.json file.

Next, import the cloudinary-core library from the cloudinary-core module:

```
let cloudinary = require("cloudinary-core"); //ES5import {Cloudinary} ;
```

Alternatively, add the JavaScript files to your HTML page:

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

To use the Cloudinary JavaScript library, you must set up configuration parameters and a cloud_name. To get a cloud_name, sign up for a free Cloudinary account, after which your cloud name is displayed in your account's dashboard. You can also specify other configuration parameters, if desired.

To set up configuration parameters while creating a new Cloudinary instance:

```
let cloudinary = new cloudinary.Cloudinary({cloud_name: "demo", secure
```

For more guidelines on Cloudinary's client-side setup, check out the documentation on integration with Cloudinary's JavaScript SDK.

Delivery of Transformations

While transforming a media asset (image or video), Cloudinary builds a delivery URL with these two parameters:

- **Action parameters,** such as b(background), which perform a specific transformation on the asset.
- **Qualifier parameters,** such as ar(aspect ratio), which alter the asset's default behavior.

You'll see many of those parameters as you proceed with this tutorial.

Image Delivery

You deliver images with either of these two directives:

- **imageTag.** This method creates an instance of the ImageTag class. You can then generate an HTML tag with the toHtml method, like this:

```
let tag = cloudinary.imageTag("sample");  tag.toHtml();
```

The code above generates this HTML image tag:

```
<img src="https://res.cloudinary.com/demo/image/upload/sample">
```

- **Direct URL.** To create an image URL instead of an HTML tag, use the url method, like this:

```
let tag = cloudinary.url("sample.jpg");
```

- The code above returns this string:

```
https://res.cloudinary.com/demo/image/upload/sample.jpg
```

Video Delivery

You deliver videos with either of these two directives:

- **videoTag.** You can specify transformations with videoTag , which automatically generates an HTML5 video tag, including the URL sources for the major formats supported by web browsers (WebM, mp4, and ogv).

```
cloudinary.videoTag("watchme", {secure: true, controls: true}).transfo
.overlay("text:arial_60:watchme").gravity("north").y(20).toHtml()
```

- **Direct URL**:

```
let tag = cloudinary.video_url("dog.mp4");
```

The code above returns this string:

```
https://res.cloudinary.com/demo/video/upload/dog.mp4
```

The sections below describe how to transform images and videos with **direct URLs.**

Image Transformations

Below are five common image transformations by way of Cloudinary.

1. Transform an image to a thumbnail

A **thumbnail** is a smaller version of a digital image. With Cloudinary ‘s Transformation URL API, you can create thumbnails from either face-detection or custom coordinates. Configure the c_thumb action parameter alongside the g(gravity) qualifier parameter set to one of the face detection or g_custom transformation values, for example, g_face.**Use case**: This transformation can automatically crop an image uploaded for a user profile in an app.

Example: Crop an image to a 200×200 thumbnail through face detection (g_face,c_thumb,w_200,h_200).

Here’s the original image:

```
cloudinary.url("man.jpg", {gravity: "face", width: 200, height: 200, cr
```

The code above returns this string:

```
https://res.cloudinary.com/demo/image/g_face,c_thumb,w_200,h_200/uploa
```

Here's the cropped thumbnail.

For more details, check out Cloudinary's documentation on creating thumbnails.

2. Transform an image to enhance its color accessibility

The need for **accessibility** (**a11y**) is universally recognized and supported. As a web developer, you must ensure that your software is accessible to all, including those with disabilities or impairments. With the e_assist_colorblind[:

<assist type>] transformation effect, you can apply stripes or color adjustments to help color-blind people differentiate between colors that look similar to them.

Example: Convert the red and green colors of the image below to ones that are easier to differentiate (e_assist_colorblind:xray).
Here's the original image:


```
cloudinary.url("red-green-text.png", {effect: "assist_colorblind:xray"
```

The code above returns this string:

```
https://res.cloudinary.com/demo/image/upload/e_assist_colorblind:xray/
```

Here's the live color-accessible image.

For more details on how to enable color accessibility with Cloudinary, read this article: *Open Your Eyes to Color Accessibility*.

3. Insert a placeholder if a requested image does not exist

The d_<image asset> parameter delivers a back-up placeholder if the image requested does not exist.

Use case: If a user does not upload a profile picture, Cloudinary displays the placeholder image.

Example: Return the image with the avatar public ID as a PNG if the requested image in the URL non_existing_id does not exist (d_avatar.png).

```
cloudinary.url("non_existing_id.png", {defaultImage: "avatar.png"})
```

The code above returns this string:

```
https://res.cloudinary.com/demo/image/upload/d_avatar.png/non_existing_
```

Here's the live placeholder image.

4. Make an image look artistic

With the e_art:<filter> transformation effect, you can apply a long list of filter values, e.g., incognito, al_dente, athena, fes, frost, hairspray, etc.

Use case: This transformation yields an artistic look for your image, saving you time and effort spent on photo-editing tools.

Example: Apply the incognito artistic filter (e_art:incognito).
Here's the original image:

```
cloudinary.url("medieval.jpg", {effect: "art:incognito"})
```

The code above returns this string:

```
https://res.cloudinary.com/demo/image/upload/e_art:incognito/medieval.
```

Here's the live artistic image.

For more details, see Cloudinary's documentation on artistic filter effects.

5. Smartly scale images

You can smartly scale images with Imagga, which offers two smart-cropping modes: scale and crop. The scale mode, which is similar to Cloudinary's standard fill mode, generates an image with the exact width and height dimensions you specify. To try out Cloudinary's Imagga add-on for free, first sign up for a free Cloudinary account. Later on, you can subscribe to an Imagga add-on plan that best matches your usage requirements.

Here's the original image:

```
cloudinary.url("family.jpg", {height: 240, width: 260, crop: "imagga_s
```

The code above returns this string:

```
https://res.cloudinary.com/demo/image/upload/c_imagga_scale,h_240,w_26
```

Here's the live, smartly-scaled image.

For more details, see Cloudinary's documentation on Imagga crop and scale.

Video Transformations

Below are five common video transformations by way of Cloudinary.

1. Make a video boomerang

You can make a video clip play forward and then backward with the e_boomerang parameter. Furthermore, you can deliver a classic (short, repeating) boomerang clip by specifying one or more of the trimming parameters (duration, start_offset, end_offset, and the loop effect).

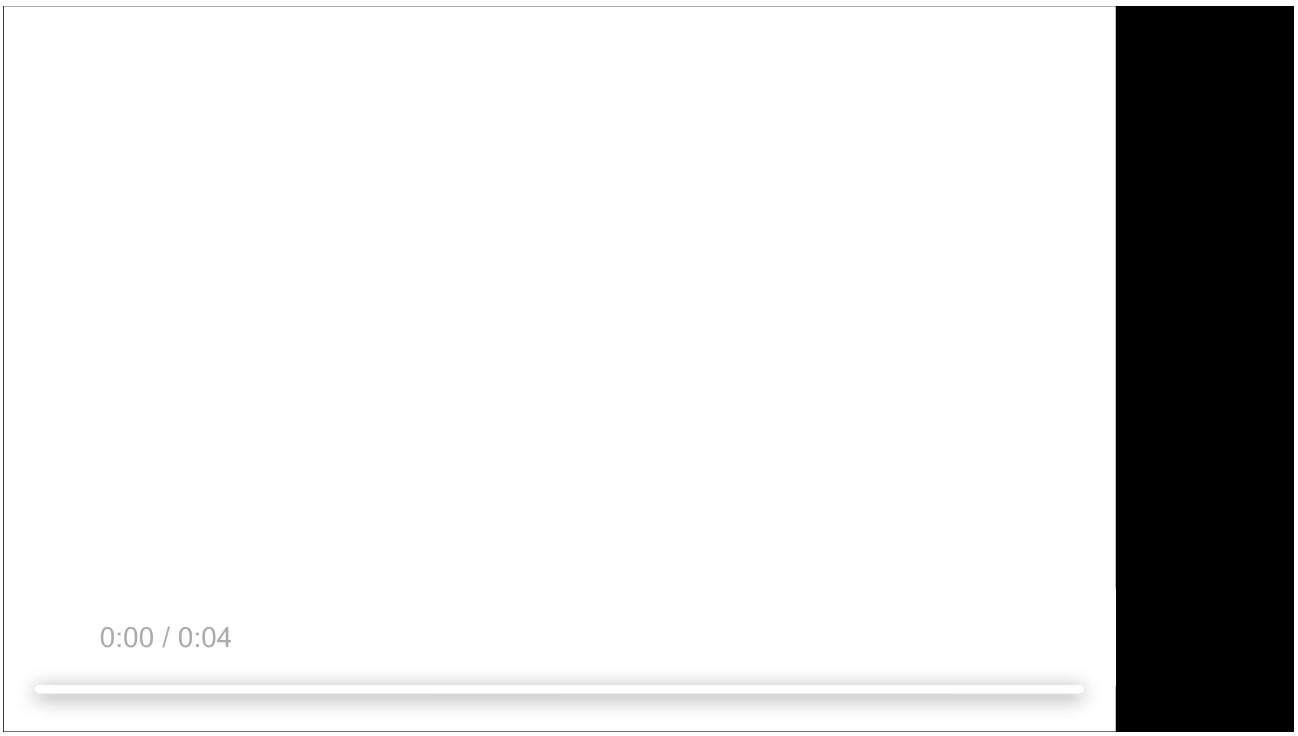
Use Case: Boomerang videos are common in ads.

Example:

```
cloudinary.video_url("dog-running.mp4",{transformation: [  {startOffse
```

The code above returns this string:

```
https://res.cloudinary.com/demo/video/upload/so_3,eo_5,e_boomerang/dog
```



Here's the live boomerang video clip.

For more details, check out Cloudinary's documentation on creating boomerang video clips.

2. Create transition between two videos

The e_transition qualifier, along with l_video, applies a custom transition between two concatenated videos.

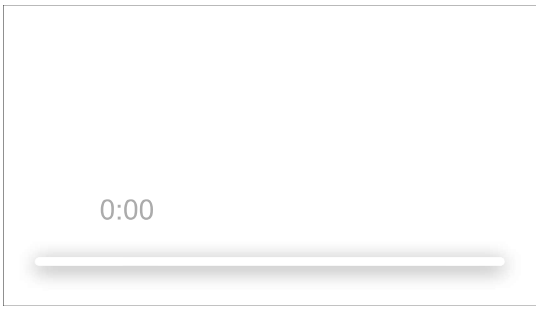
Use case: This transformation is useful for creating short video skits, ads, and the like.

Example: Apply a transition effect between two concatenated videos (l_video:transition1,e_transition):

```
cloudinary.video_url('kitten_fighting', {transformation: [  {height: 1
```

The code above returns this string:

```
https://res.cloudinary.com/demo/video/upload/c_fill,h_150/l_video:dog,
```



Here's the live video with the transformation.

For more details, see Cloudinary's documentation on concatenating videos with custom transitions.

3. Create video previews

The e_preview[:duration_<duration>][:max_seg_<max segments>][:min_seg_dur_<min segment duration>] parameters generate an excerpt of a video through Cloudinary's AI-powered preview algorithm, which creates a preview after identifying the video's most captivating segments.

Use case: Previews are popular as video trailers.

Example: Generate a 12-second video preview with a maximum of three segments and a minimum segment-duration of three seconds (e_preview:duration_12:max_seg_3:min_seg_dur_3):

```
cloudinary.video_url('short_film', {effect: "preview:duration_12:max_s
```

The code above returns this string:

```
https://res.cloudinary.com/demo/video/upload/e_preview:duration_12:max_
```

Here's the live video preview.

For more details, see Cloudinary's documentation on generating AI-based video previews.

4. Set the duration of videos

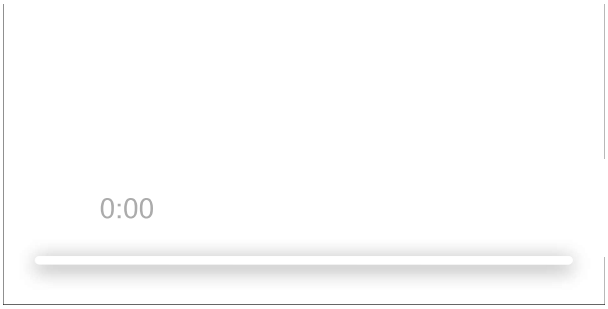
You can **trim** a video or audio clip to the length (in seconds) you specify with the du_<time value> parameter, or use it as a qualifier to control the duration of a corresponding transformation.

Example: Deliver a 30-second video (du_30):

```
cloudinary.video_url("short_film.mp4", {duration: "30"})
```

The code above returns this string:

```
https://res.cloudinary.com/demo/video/upload/du_30/short_film.mp4
```



Here's the live video with the transformation.

5. Modify video quality

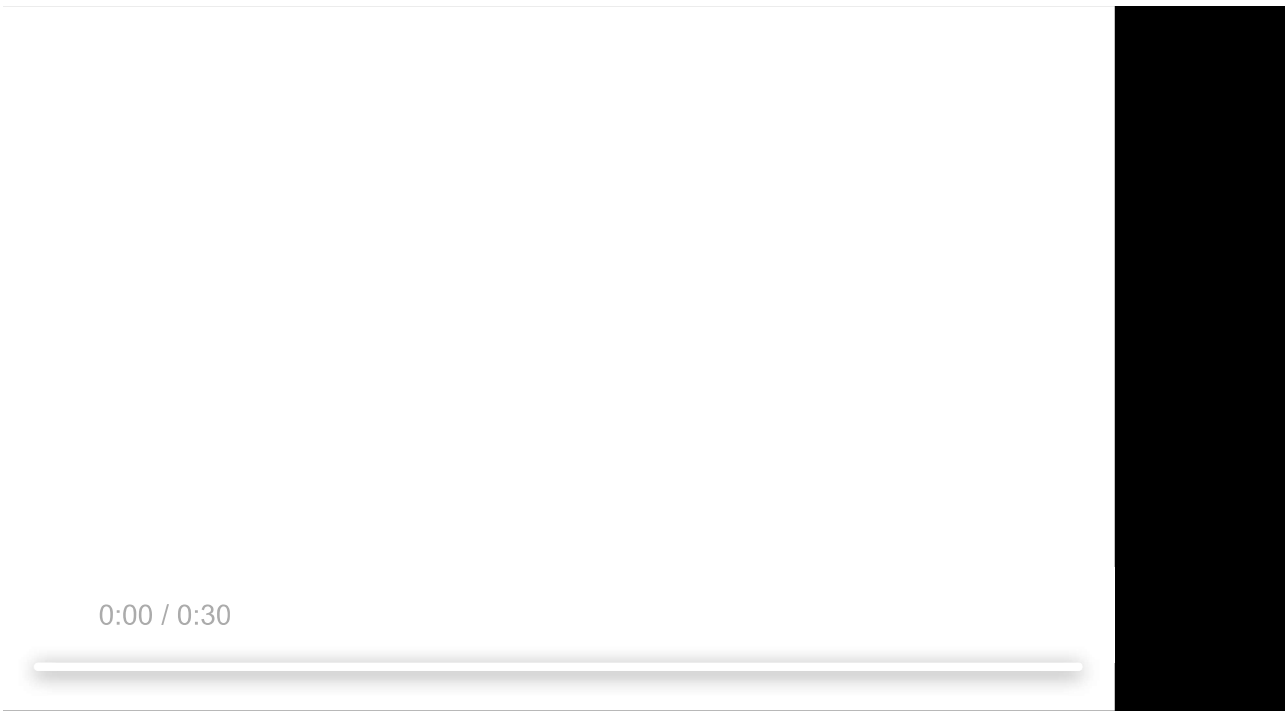
The q_<quality level>[:qmax_<quant value>][:<chroma>] parameter sets the quality to the level you specify. This transformation is especially useful for delivering videos to geographic areas with a slow Internet connection.

Example: Transcode a video to the WebM format with 70% quality, setting the maximum quantization to 20% (q_70:qmax_20):

```
cloudinary.video_url('short_film.webm', {duration: "30", quality: "70:
```

The code above returns this string:

```
https://res.cloudinary.com/demo/video/upload/du_30,q_70:qmax_20/short_
```



Here's the live video with the transformation.

For more details. see Cloudinary's documentation on modifying video quality.

Summary and Resources

Cloudinary is an excellent tool with optimization and transformation capabilities. To try out the transformations described in this tutorial, first sign up for a free Cloudinary account.

The following resources from Cloudinary are handy references:

- *Transformation URL API*

- *Image Transformation Guide*
- *Video Transformation Guide*
- *Image Delivery Guide*