AcroTeX.Net

The ltx4yt Package

Playing YouTube Videos in the default browser

D. P. Story

Prepared: July 30, 2020

www.acrotex.net
Version 0.7, 2020/07/30

Table of Contents

1. Introduction

My beautiful package yt4pdf is gone, time has caught up with it: The Flash player is no longer in use, anywhere;[1] Google discontinued its ActionScript 3.0 Player several years ago. It was Acrobat and Google technologies that were the foundation of the yt4pdf package.

We turn over a new leaf by writing and publishing the ltx4yt package: LaTeX for YouTube. What ltx4yt does is to provide some tools for creating links, dropdown lists, popup menus for playing selected YouTube videos in the default browser. Perfect for personal use and for academic, professional, or classroom presentations that refer to YouTube content.

1.1. Sample files

The following are the sample files shipped with ltx4yt:

- `ltx4yt-1.tex` demonstrates links created by `\ytvId` and `\ytLink`. Such links created in a PDF are pretty much functional for all PDF viewers, including the native PDF viewers of all the modern browsers, and PDF viewers on other devices such as Android and iPhones.
- `ltx4yt-2.tex` demonstrates the PDF forms controls related methods of selecting a video to watch. The command pair `\ytComboList` and `\ytComboBtn` provide support for creating a dropdown list (combo box) for selecting a video by title, the push button takes the selection and passes it to YouTube for display in the default browser. The sample also includes a popup menu for selecting video by title. Finally, there is a query field in enter query text and to search YouTube for all related titles.

  This file (`ltx4yt-2.pdf`) performs as designed in Acrobat, Adobe Reader DC, and PDFX-Change Editor, and *will not* be functional in native PDF viewers of browsers. `ltx4yt-2.pdf` will have no functionality in PDF viewers on hand-held devices such as smart phones or tablets.

2. Options, Requirements, and Workflows

usepopup option

Options. Currently, this package has two options: `usepopup` and `!usepopup`. When `usepopup` is taken, additional commands are defined and supporting document JavaScript is embedded in the PDF. See Section 3.3 on page 7 for details on using popups in the document. The `!usepopup` is a convenience option that turns off the `usepopup` option. The default is to not use popups in the document (`!usepopup`).

Requirements. This package requires the eforms package, which is part of acrotex.[2] If the option `usepopup`, the package popupmenu is also input.[3]

[1] Adobe will formally abandon Flash Player in December 2020
[2] http://www.ctan/pkg/acrotex
[3] http://www.ctan/pkg/popupmenu

Workflows. This is a general LaTeX package, any workflows can be used to build a ltx4yt document: pdflatex, lualatex, xelatex, or dvips -> distiller.[4]

3. Basic Methods: Playing a YouTube video using its Video ID

There are three methods of playing a YouTube video based on its video ID:

- **Using links**
- **Using dropdown lists**
- **Using a popup menu**

In each of the subsequent sections, we discuss each of these methods in turn.

3.1. Basic Methods: Links with \ytvId

Demo files. The techniques of this section are illustrated by the two files `examples/ltx4yt-1.tex` and `examples/web-pkg/ltx4yt-w1.tex`.

The underlying command that creates links is the eforms command \setLink, this command is not seen, but is part of all link commands in this section.

To create a link that references a YouTube video from its video ID, use the \ytvId command.

```
\ytvId*[⟨KV-pairs⟩]{⟨ytvID⟩}{⟨text⟩}
\ytvIdPresets{⟨KV-pairs⟩}
```
(1)

Parameter Description: Use the optional asterisk (*) when the referenced video *cannot be embedded*; *not specifying* the asterisk means the video *can be embedded*. **Embedding is the best way of viewing a video.** The ⟨*KV-pairs*⟩ are key-value pairs recognized by the underlying \setLinks command. The ⟨*ytvID*⟩ is the YouTube video ID of the video to be played in the default browser. Finally, ⟨*text*⟩ is the text around of which the link is constructed.

The \ytvIdPresets command is a convenient way of setting the link options *uniformly* for all links created by \ytvId. Individual ⟨*KV-pairs*⟩ passed though the optional argument of \ytvId will override the preset values. Its default definition is,

```
\ytvIdPresets{\linktxtcolor{webbrown}}
```

This sets the color of the link to a brown color, which is why the links of this document are brown.

[4] dvips -> ps2pdf can be used if only links are used to reference YouTube videos and the !usepopup option is in force (no document JavaScript).

An example. There are two "Kung-Fu Fighting" videos of interest: GZ9e3Dy7obA and jhUkGIsKvn0; we set up likes for the first and twice for the second:

Kung-Fu Fighting (Bruce Lee)
```
\ytvId{GZ9e3Dy7obA}{Kung-Fu Fighting (Bruce Lee)}
```
Kung-Fu Fighting (Carl Douglas)
```
\ytvId{jhUkGIsKvn0}{Kung-Fu Fighting (Carl Douglas)}
```
Kung-Fu Fighting* (Carl Douglas)
```
\ytvId*{jhUkGIsKvn0}{Kung-Fu Fighting* (Carl Douglas)}
```

The first link works fine, just the video and nothing else. The second one does not, the video poster is loaded, but when you play the video, the response is Video unavailable. The third link works fine now. The problem here is that the second video cannot be embedded; we have to use the *-option to create a different link, one that goes to the full YouTube web site to see one or more advertisements before the video is allowed to be played, additionally, there are numerous extraneous content everywhere on the page. When a video *can be embedded*, you get a very clean video experience, free of advertisements and extraneous content; *this is the best case.* Try using the first link again to enjoy the experience.

There is a multi-line version of \ytvId,

```
\ytvIdML*[⟨KV-pairs⟩]{⟨ytvID⟩}{⟨text⟩}
\ytvIdPresets{⟨KV-pairs⟩}
```
(2)

This form, which requires aeb_mlink package and the dvips->distiller, creates true multi-line links. Here is an example: Lori's Corner: Episode #1, I met her back in the year 2000, though she probably does not remember. The applications pdflatex, lualatex, and xelatex will break this link cross lines, but does so by creating two links. This link is a single link that inverts across both lines when clicked. \ytvIdML uses the same presets command \ytIdPresets.

• How to get the YouTube video ID

Go to the YouTube web site and search for a video of interest. Once found, play, then pause the video. Go the browser's URL address bar and you'll see something like this:

https://www.youtube.com/watch?v=dAgfnK528RA
(ytvId)

Copy the ytvId into the first argument of \ytvId and supply a title:

Math Antics - Order of Operations
```
\ytvId{dAgfnK528RA}{Math Antics - Order of Operations}
```

Cool! You can also get the video ID by right clicking on the paused video and exploring the context menu presented.

• **Can the video be embedded?**

I've done a little research on this question, the answer is that there is no way of knowing in advance. You need to test each link. In the case of video ID dAgfnK528RA, we found it, we created the link with \ytvId, we tested it, and it worked! Had it not worked (the dreaded Video unavailable appears when you try to play the video), we simply use the *-option. It's just that simple.

3.2. Basic Methods: dropdown list

Demo files. The techniques of this section are illustrated by the two files examples/ltx4yt-2.tex and examples/web-pkg/ltx4yt-w2.tex.

A dropdown list, a combobox in Adobe's original terminology, is an Adobe form field that drops down to display a list of menu items. It takes up less page space, and may be a good choice rather than listing a number of links on the page.

To create a dropdown list

1. **Create a play list.** At any point prior to the dropdown list, declare your play list:

```
\declarePlayList{⟨\plCmd⟩}{%
  \ytIdTitle{⟨title1⟩}{⟨ytvId1⟩}
  ...
  \ytIdTitle{⟨titlen⟩}{⟨ytvIdn⟩}
}
```

*use * in title*

If the video referenced by ⟨*ytvId*⟩ *cannot be embedded*, then place an * in the ⟨*title*⟩ as a single to the underlying JavaScript that the video cannot be embedded.

2. **Set the initial value and playlist** At any point prior to the dropdown list, set the require dropdown list information:

```
\ytPlayList{⟨ytvId⟩}{⟨\plCmd⟩}
```

where ⟨*ytvId*⟩ is the video ID of the title initially displayed in the dropdown list, and ⟨*\plCmd*⟩ is a command defined earlier in a \declarePlayList command.

3. **Place the dropdown list and accompanying Play button.**

```
\ytComboList[⟨KV-pairs⟩]{⟨name⟩}{⟨wd⟩}{⟨ht⟩}
\ytComboBtn[⟨KV-pairs⟩]{⟨name⟩}{⟨wd⟩}{⟨ht⟩}
```

where ⟨*name*⟩ is a text string of ASCII letters and numbers. The ⟨*name*⟩ is appended on to the field names of these two field; The pair \ytComboList and \ytComboBtn must be passed the same ⟨*name*⟩ value. The common ⟨*name*⟩ ties them together.

Below is an abbreviated example, more extensive examples are found in the demo files sited at the beginning of this section.

Example A four-item dropdown list: Kung-Fu Fighting (Bruce Lee) PLAY

The verbatim listing is,

```
\declarePlayList{\playList}{% note: put parentheses within braces
  \ytIdTitle{Kung-Fu Fighting {(Bruce Lee)}}{GZ9e3Dy7obA}
  \ytIdTitle{Kung-Fu Fighting* {(Carl Douglas)}}{jhUkGIsKvn0}
❶   \ytIdTitle{J\"{u}rgen's "favorite" song*}{mLDF5MBMWHE}
❷   \ytIdTitle{Learn \cs{LaTeX} in one video}{VhmkLrOjLsw}
}
\ytPlayList{GZ9e3Dy7obA}{\playList}
\paragraph*{Example} A two-item dropdown list:
  \ytComboList{YT1}{144bp}{11bp}\olBdry\ytComboBtn{YT1}{33bp}{11bp}
```

There are a several observations to make in this markup: line ❶ uses standard LaTeX markup to describe the u-umlaut, the double quote requires no special attention; line ❷ the backslash can be expressed as `\cs{`⟨*text*⟩`}`. Also note that two of the titles have an * in them, this signals that these videos cannot be embedded.

3.3. Basic Methods: popup menus

Demo files. The techniques of this section are illustrated by the two files `examples/ltx4yt-2.tex` and `examples/web-pkg/ltx4yt-w2.tex`.

A pop-up menu is a menu list that appears on top of the content of the page, it is generated by the JavaScript method `app.popUpMenuEx()`. The advantage of this method is that it take up no space on the page, it is displayed on top of the page and is dismissed when an item from the menu list is chosen.

To create a pop-up menu

1. **Build a menu listing.** Use the popupmenu environment of the popumemnu package to build/design your menu.

```
\begin{popupmenu}{⟨menu-name⟩}
\puIdTitle{⟨title1⟩}{⟨ytvId1⟩}
...
\puIdTitle{⟨titlem⟩}{⟨ytvIdm⟩}
\begin{submenu}{title=⟨submenu-name1⟩}
  \puIdTitle{⟨titlem+1⟩}{⟨ytvIdm+1⟩}
  ..
  \puIdTitle{⟨titlem+n⟩}{⟨ytvIdm+n⟩}
\end{submenu}
...
\end{popupmenu}
```

You need not have a submenu structure, but submenus are useful for organizing the links. The first argument of `\puIdTitle` is passed through `\pdfstringdef`, this enables you to use LaTeX markups for accents, for example. If the second argument

(⟨*ytvId*⟩) is empty, then that entry has no action associated with it, and can be used as a menu heading.

2. **Create the menu data.** Following all popupmenu environments, yet still in the preamble, insert the following command:

preamble

```
\ytUseMenus{⟨menu-name1⟩,...,⟨menu-namek⟩}
...
\begin{document}
```

The argument of \ytUseMenus is a comma-delimited list of ⟨*menu-name*⟩s declared as the first argument of popupmenu environments.

3. **Place the \ytPopupBtn command.**

```
\ytPopupPresets{⟨KV-pairs⟩}
\ytPopupBtn[⟨KV-pairs⟩]{⟨menu-name⟩}{⟨wd⟩}{⟨ht⟩}
```

Employ \ytPopupPresets as a way to pass ⟨*KV-pairs*⟩ to all \ytPopupBtn commands in the document. For an individual \ytPopupBtn command, its ⟨*KV-pairs*⟩ argument passes key-values to that \ytPopupBtn, these ⟨*KV-pairs*⟩ will override the ones of \ytPopupPresets. The second argument, ⟨*menu-name*⟩, is name of the pop-up menu data you want to use. This menu data is created earlier in the preamble by the popumenu environment and referenced again in the argument of \ytUseMenus.

Example. Sea Hunt Let's create a short pop-up menu, more extensive examples are found in examples/ltx4yt-2.tex and examples/web-pkg/ltx4yt-w2.tex.

```
   \begin{popupmenu}{YTSea}
❶    \puIdTitle{J\"{u}rgen's favorite song*}{mLDF5MBMWHE}
❷    \puIdTitle{\Esc"Sea Hunt\Esc"
       US TV series {(1958-61)} lead-in}{Lz0aMoWh8Q4}
     \puIdTitle{Theme Song to Sea Hunt*}{2QxXk6X9GDo}
❸    \puIdTitle{Learn \cs{LaTeX} in one video}{VhmkLrOjLsw}
   \end{popupmenu}
❹  \ytUseMenus{YTSea}
   ...
   \begin{document}
   ...
   ...
   \paragraph*{Example.}
   \ytPopupBtn[\CA{Sea Hunt}]{YTSea}{20bp}{5bp} Let's ...
```

The first argument of \puIdTitle is passed through \pdfstringdef, this enables you to use LaTeX markup on accents, as in ❶. In line ❷, a special locally defined command \Esc is used to "escape" the double quotes so that we \" to appear in the document JavaScript, which is where the menu items appear within the PDF. In line ❸ a special

locally defined command \cs is used to create a backslash, in a manner similar to LaTeX markup. Note the presence of the * in the title of two of the menu items; this is used to signal to the underlying JavaScript that the title cannot be embedded. Finally, we the popupmenu environment, we declare, in line ❹, that we are using the menu named YTSea.

** noted*

4. Advanced Methods: Playing video with \ytLink

The link earlier discussed is \ytvId, which takes as its first argument the video ID of the targeted video. It then constructs a URL one way or another, depending on the presence of the *-option. This method is pretty rigid, one size fits all, if you will. A more flexible method of constructing links is to use the \ytLink command.

```
\ytLink{\embedId{⟨ytvId⟩}\params{⟨params⟩}}{⟨text⟩}      ❶
\ytLink{\watchId{⟨ytvId⟩}\params{⟨params⟩}}{⟨text⟩}      ❷
\ytLink{\embed{⟨spec⟩}}{⟨text⟩}                          ❸
\ytLink{⟨spec⟩}{⟨text⟩}                                  ❹
\ytLink{\channel{⟨name⟩}}{⟨text⟩}                        ❺
\ytLink{\user{⟨name⟩}}{⟨text⟩}                           ❺
\ytvIdPresets{⟨KV-pairs⟩}          (presets for \ytvId)
```

Each of these has an optional first argument ⟨*KV-pairs*⟩ that is not shown above. Let's take a look at each of these in turn and illustrate with examples. Yes, YouTube has a number of parameter is recognizes in its urls, see here for a discussion.

❶ Use this form *when you can embed* a video ⟨*ytvId*⟩ with additional parameters. (this best type of video). The \params argument *must follow* the \embedId, its arguments are any parameters you want to append to the URL. The use of \params is optional; however, without \params you should use \ytvId*{⟨*ytvID*⟩}{⟨*text*⟩}. For example, we auto play a video with modest branding:

Lori's Corner: Episode #1

```
\ytLink{\embedId{5y9-EVmreU4}
   \params{autoplay=1&modestbranding=1}}{Lori's Corner: Episode \#1}
```

The demo file ltx4yt-1.tex contains an example of this type of link that sets up a small play list of three videos.

❷ Use this form *when you* cannot *embed* (or you want to play the video on the main YouTube site) ⟨*ytvId*⟩ with additional parameters. The \params argument must follow the \watchId, its argument are any parameters you want to append to the URL. The use of \params is optional; however, without the \params you may as well use \ytvId*{⟨*ytvID*⟩}{⟨*text*⟩}.

Lori's Corner: Episode #1

```
\ytLink{\watchId{5y9-EVmreU4}
   \params{autoplay=1}}{Lori's Corner: Episode \#1}
```

❸ A more general form that allows you to formulate general URLs. For example, we *search* for YouTube videos on Adobe Acrobat DC:

Search for Adobe Acrobat DC

```
\ytLink{\embed{listType=search&list=Adobe Acrobat DC}}
   {Search for Adobe Acrobat DC}
```

❹ The most general form, ⟨*spec*⟩ is simply appended, ie, `https://www.youtube.com/`⟨*spec*⟩

Totally custom search link
```
\ytLink{embed?listType=search&list=LaTeX typesetting}
   {Totally custom search link}
```

❺ There are a couple of URLs for displaying a channel on YouTube, at least there are a couple that I have discovered. Use this form to see channel of a particular contributor to YouTube. For example,

The RocketJump Channel
```
\ytLink{\channel{rocketjump}}{The RocketJump Channel}
```
Freddie Diew's Channel
```
\ytLink{\user{freddiew}}{Freddie Diew's channel}
```

To get the true ⟨*name*⟩ of a channel, go to YouTube and search for that channel. For example, search for RocketJump. On the resulting page, click on a RocketJump link and look at the location bar in the browser to see the path, it may come up `c/rocketjump` or `user/freddiew`.

There is a multi-line version of `\ytLink`.

```
\ytLinkML*[⟨KV-pairs⟩]{⟨spec⟩}{⟨text⟩}
\ytvIdPresets{⟨KV-pairs⟩}
```
(3)

This form, which requires aeb_mlink package and the dvips->distiller, creates true multi-line links. `\ytLinkML` has the same variations of `\ytLink` (lines ❶–❺) described earlier

5. Searching YouTube

Want to interactively search YouTube for your favorate video? Search using the commands `\ytInputQuery` and `\ytSearch`:

```
\ytInputQuery[⟨KV-pairs⟩]{⟨wd⟩}{⟨ht⟩}
\ytSearch[⟨KV-pairs⟩]{⟨wd⟩}{⟨ht⟩}
\ytClearQuery[⟨KV-pairs⟩]{⟨wd⟩}{⟨ht⟩}
```

Search Clear

`\ytInputQuery` is an input box to enter query text; `\ytSearch` is a push button that searchs YouTube for the text entered into `\ytInputQuery`; and finally, `\ytClear` clears `\ytInputQuery`.

That's all for now, I simply must get back to my retirement. 🕮