CDK Documentation

Release v0.0.8

Simeon Franklin

Sep 27, 2017

Contents

Quickly create code-heavy html slide decks from plain text.

CHAPTER 1

Start Here

It's easy to get started with *cdk*. You can be viewing your nicely formatted slides in less than a minute!

Installing *cdk*

cdk can be installed with standard Python packaging tools. Even if you are not a Python developer you likely have them installed on Linux or OSX systems.

Using *pip*

Just run

```
$ sudo pip install cdk
```

Using *easy_install*

If you don't have *pip* you may have an older tool called *easy_install*. Try

```
$ sudo easy_install cdk
```

Manual Installation

If you don't have any standard Python installation tools available you can download the latest installation bundle from https://pypi.python.org/pypi/cdk/1.0.8

Unzip the tarball, cd insto the resulting directory and run:

```
$ python setup.py install
```

Upgrading an existing installation

If you already have *cdk* installed you can upgrade to the latest version at any time by running:

```
$ sudo pip install --upgrade cdk
```

Using *cdk*

cdk transforms asciidoc documents into single file html slide decks.

Generating a Sample Deck

You can generate a simple slide deck with

```
$ cdk --generate=sample.asc
```

The sample file might look something like:

```
= My Presentation

Author: @somebody

== My First Slide

Presentation software for engineers

[options="incremental"]
* Content should be simple but presentation stylish.
* Author slides in plain text (asciidoc)
* And play them in the browser.

== My Second Slide

Use the source!

[source,python]
----
>>> x = list("Simeon")
>>> x.lower()
"simeon"

----
```

Building your Slides

Now run *cdk* on your input file to produce a single html file you can open in your browser.

```
$ cdk -o sample.asc
```

Whihc will open the resulting *sample.html* file in your browser! Hit the space bar to advance a slide or hit the "h" key to see what other commands are available to navigate through the deck.

Learning Asciidoc

You'll want to learn more about asciidoc - it doesn't take much effort to learn to make links, bulleted lists, header levels, code samples and more.

I frequently recommend the Asciidoc cheat sheet - available at http://powerman.name/doc/asciidoc

CHAPTER 2

Learn More

If you're interested in learning more about *cdk* or the underlying technologies read on.

About

tl;dr - using *cdk* write documents in asciidoc and produce elegant single-file slidedecks as html.

cdk is a Python program to more easily stitch together Asciidoc and asciidoc-deckjs which allows asciidoc to create deck.js slide decks. This means you can author presentations in plain text in a markdown-like format and get well behaved attractive presentations in a single html file.

In addition *cdk* is opinionated. It uses many plugins from the *deck.js* ecosystem and provides a few custom plugins as well. It provides theming support. In general features are oriented around the kinds of things technical instructors need to do:

- showing and explaining lots of code
- organizing and navigating large slide decks
- minimizing the effort needed to make things look good

Installing/customizing Themes

cdk has a themes directory that contains the default theme named "plain".

A theme is just a directory with a .css file of the same name. For example a theme "foo" is just a directory named *foo* with a *foo.css* file in it.

cdk comes with a command to install a theme. All this means is putting a directory in a zip file - the contents will be extracted to the *cdk* theme directory. If you name your theme "plain" it will overwrite the default (but don't do that!)

cdk also comes with a flag to pick a theme for a build or to set the default to always use a theme when building a slide deck.

So basically - you need to make a .zip with a theme. Let's say you called the theme "newcircle". You could install it and make it the default with:

```
$ cdk --install-theme=newcircle.zip
$ cdk --default-theme=newcircle
```

Note: In practice I may use *lessc* and some helper scripts to make editing .css easier and to allow packing of images into the .css. See the included twitter theme at https://github.com/twitter/cdk/tree/master/cdk/custom/deck.js/themes/twitter

For another example theme contributed by a third-party check out https://github.com/thenewcircle/cdk-theme

Setting up the Development Environment

Clone the repo:

```
$ git clone https://github.com/twitter/cdk.git
```

cd into the repo directory. The *cdk/data* directory is empty - run

```
$ make getdata
```

to download the many non-pip-installable external dependencies.

I recommend working in a virtual environment while modifying the code. You can activate your virtualenv and the install the requirements in the accompanying *requirements.txt* file:

```
(cdk-virtualenv) $ pip install -r requirements.txt
```

Once you've installed the requirements and downloaded the additional data, *cdk/__init__.py* is the setuptools entry point. You can run

```
(cdk-virtualenv) $ python cdk/__init__.py --generate=sample.asc
(cdk-virtualenv) $ python cdk/__init__.py -o sample.asc
```

to create a basic deck in asciidoc and compile the slide deck.

Other Resources

Slides are asciidoc so be sure to read the docs or at least a cheatsheet.

The resulting slide deck is provided by *deck.js* using the *asciidoc-deckjs* backend so read up you might want to read the asciidoc-deckjs tutorial

Design Justifications

I'm having some challenges writing a utility that mainly stitches together other programs. In particular my main goals are:

- ease of installation. I want to be able to tell people to just *sudo easy_install cdk*
- My dependencies are not dependable: *deck.js* isn't a python package at all and *asciidoc* isn't setup to be installed.
- I don't want to have a source tree full of thousands of files I'm not authoring.

Complicating things is the difficulty of getting setuptools to include non-python data files in a known good location. Currently I am:

- including a couple of package data directories *cdk/data* and *cdk/custom*. I'm not checking *cdk/data* into source control, the contents can be recreated by running *make getdata*.
- I'm using the *MANIFEST.in* file to include all the contents of *cdk/data* and *cdk/data* in the *sdist* I'm building with *python setup.py sdist*.