
Hello
World

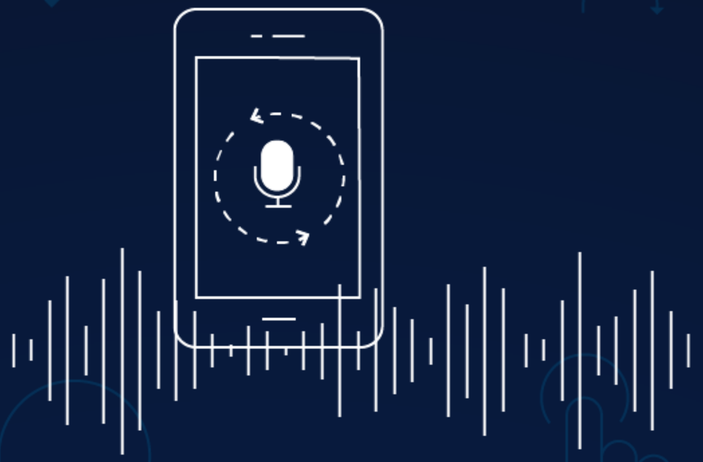

@AARONBASSETT


@AARONBASSETT

nexmo®

The Vonage® API Platform









Michael Heap
Developer Advocate
@mheap



Steve Crow
Developer Advocate
@cr0wst



Martyn Davies
Developer Advocate
@martynd



Amanda Folson
Developer Relations Manager
@AmbassadorAwsum



Lorna Mitchell
Developer Advocate
@lornajane



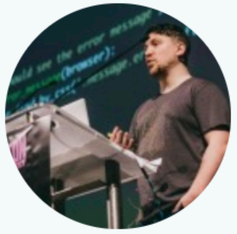
Olia Dukova
Community Manager
@oliadukova



Alex Lakatos
Developer Advocate
@lakatos88



Mark Smith
Developer Advocate
@judy2k



Aaron Bassett
Developer Advocate
@aaronbassett



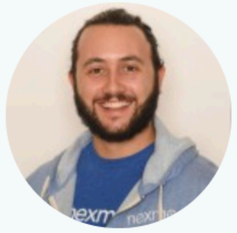
Phil Leggetter
Director of Developer Relations
@leggetter



Rabeb Othmani
Developer Advocate
@Rabeb_Othmani



Britt Barak
Developer Advocate
@brittBarak



Chris Guzman
Developer Advocate
@speaktochris



Tony Bedford
Technical Writer



Sam Machin
Developer Advocate
@sammachin



Michael Heap
Developer Advocate
@mheap



Steve Crow
Developer Advocate
@cr0wst



Martyn Davies
Developer Advocate
@martynd



Amanda Folson
Developer Relations Manager
@AmbassadorAwsum



Lorna Mitchell
Developer Advocate
@lornajane



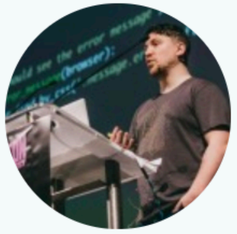
Olia Dukova
Community Manager
@oliadukova



Alex Lakatos
Developer Advocate
@lakatos88



Mark Sm
Developer Advocate
@judy2k



Aaron Bassett
Developer Advocate
@aaronbassett



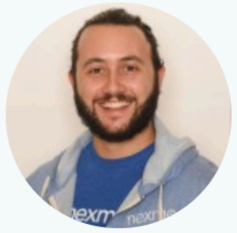
Phil Leggetter
Director of Developer Relations
@leggetter



Rabeb Othmani
Developer Advocate
@Rabeb_Othmani



Britt Barak
Developer Advocate
@brittBarak



Chris Guzman
Developer Advocate
@speaktochris



Tony Bedford
Technical Writer



Sam Machin
Developer Advocate
@sammachin

(No Model.) 2 Sheets—Sheet 1.

C. P. FEST.

AIR SHIP.

No. 263,397. Patented Aug. 29, 1882.

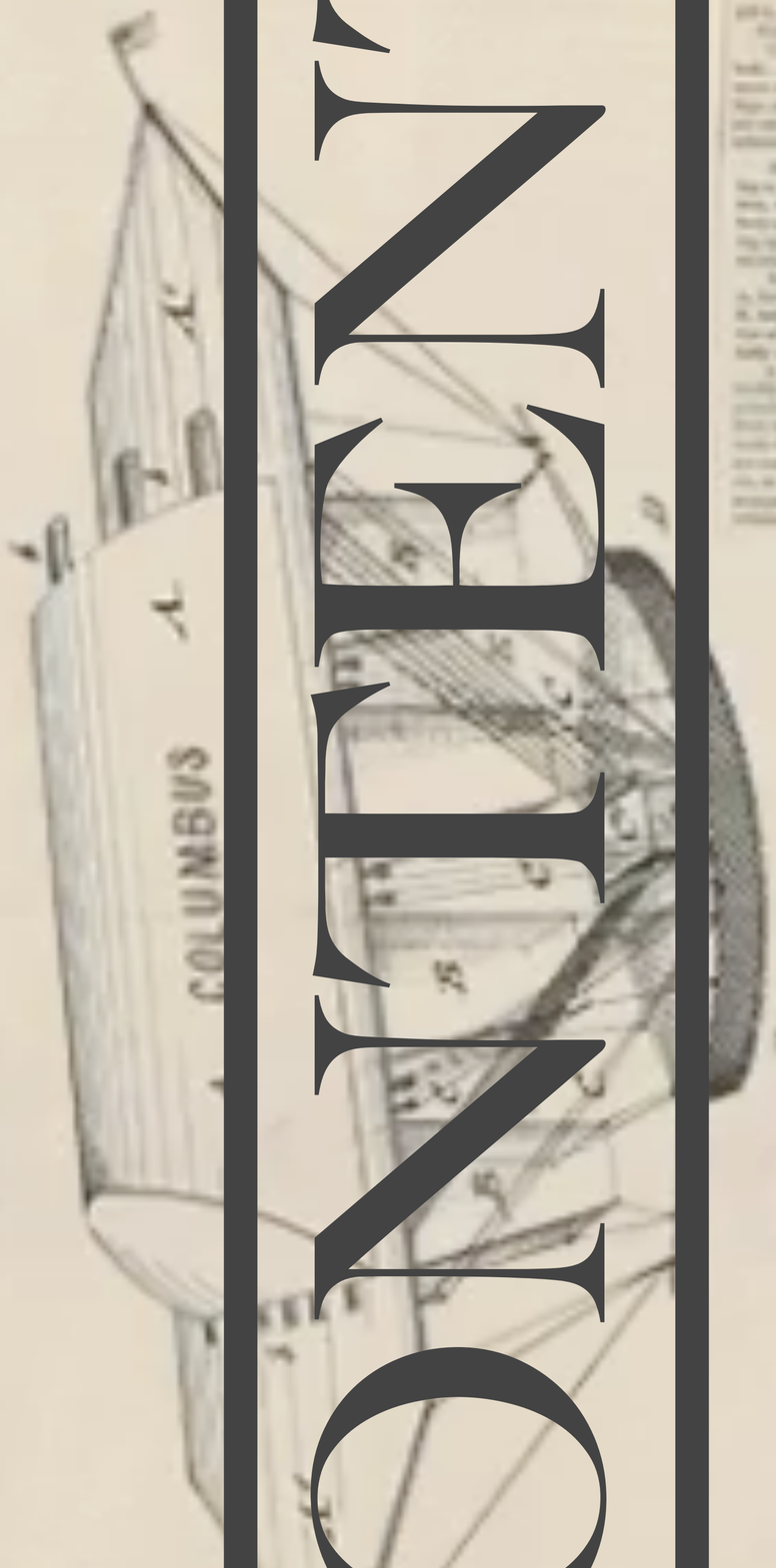
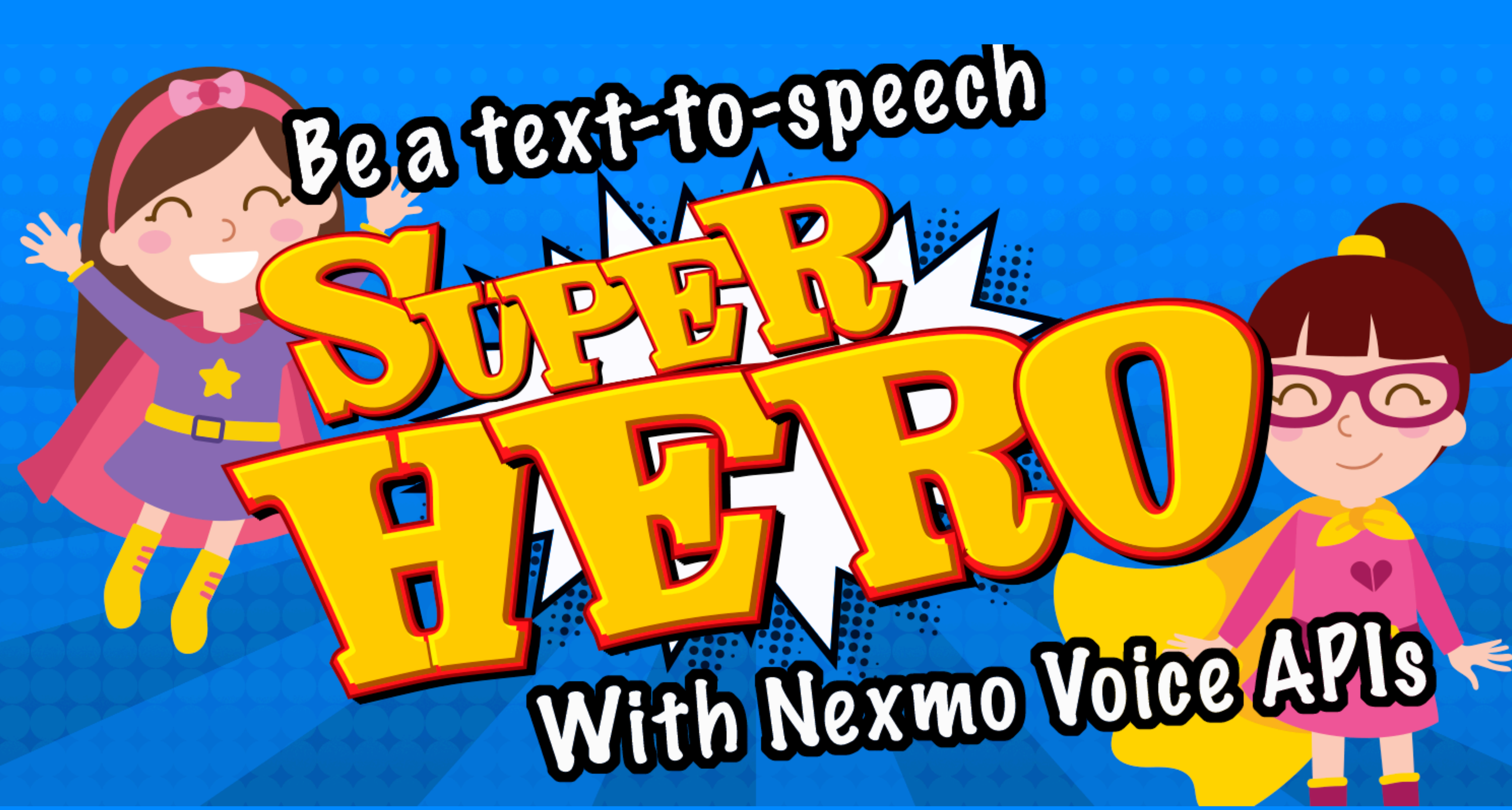


Fig. 1.

WITNESSES:

INVENTOR.

ATTORNEY


Be a text-to-speech
SUPER HERO
With Nexmo Voice APIs


CLASSIFY
AUDIO
WITH NLP

PROTECTING YOUR USER'S PRIVACY WITH VOICE PROXY





creating a

P2P Voice Journal

with Nexmo Voice APIs


python
Voice Broadcast
WITH
PYTHON
&
SANIC
ALERT

GETTING OUT OF A BAD DATE

(WITH FITBIT & NEXMO)





nexmo® | The Vonage® API Platform

fitbit®



Build a chat application with Angular Material and the In-App Messaging JavaScript SDK



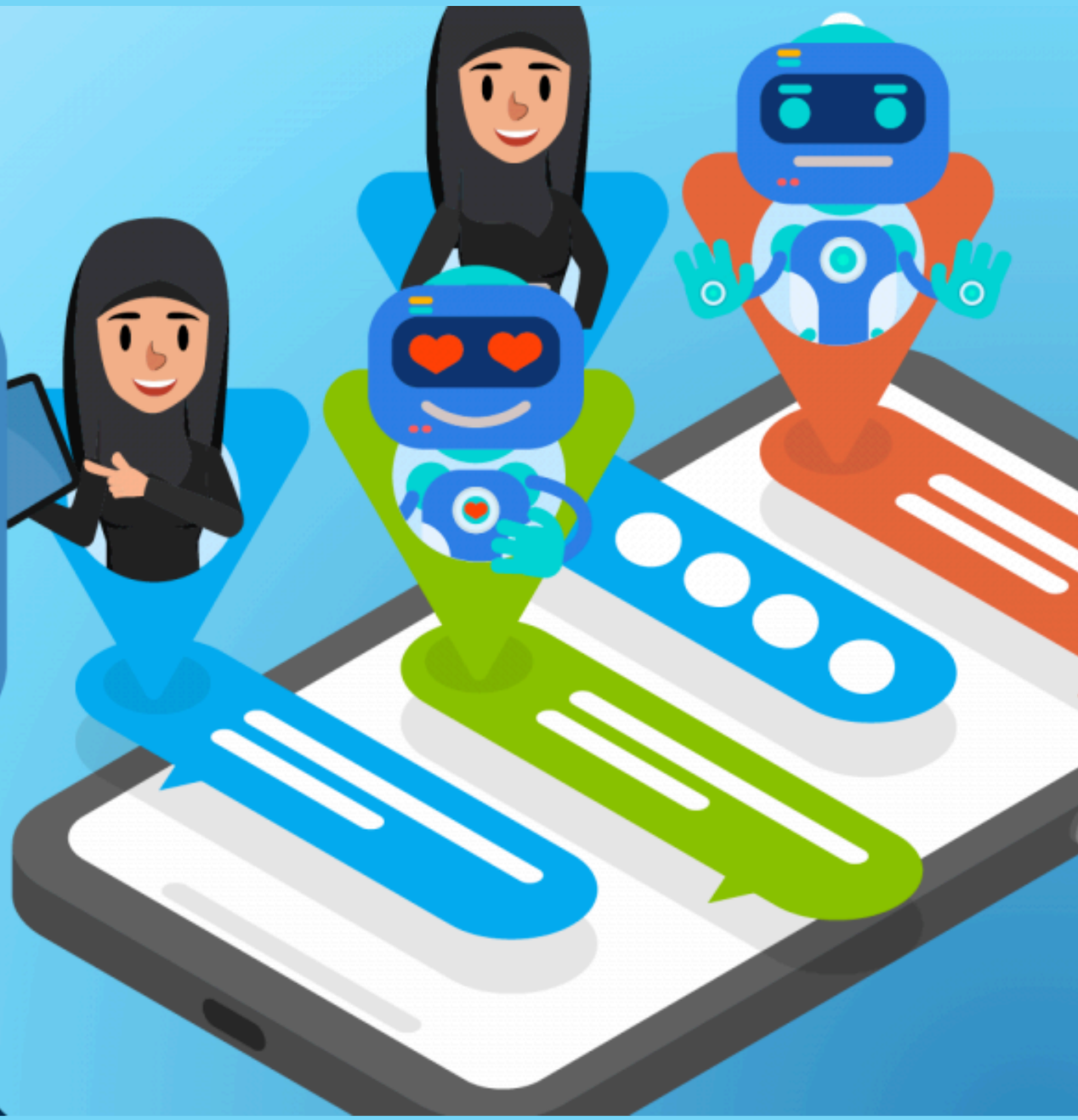


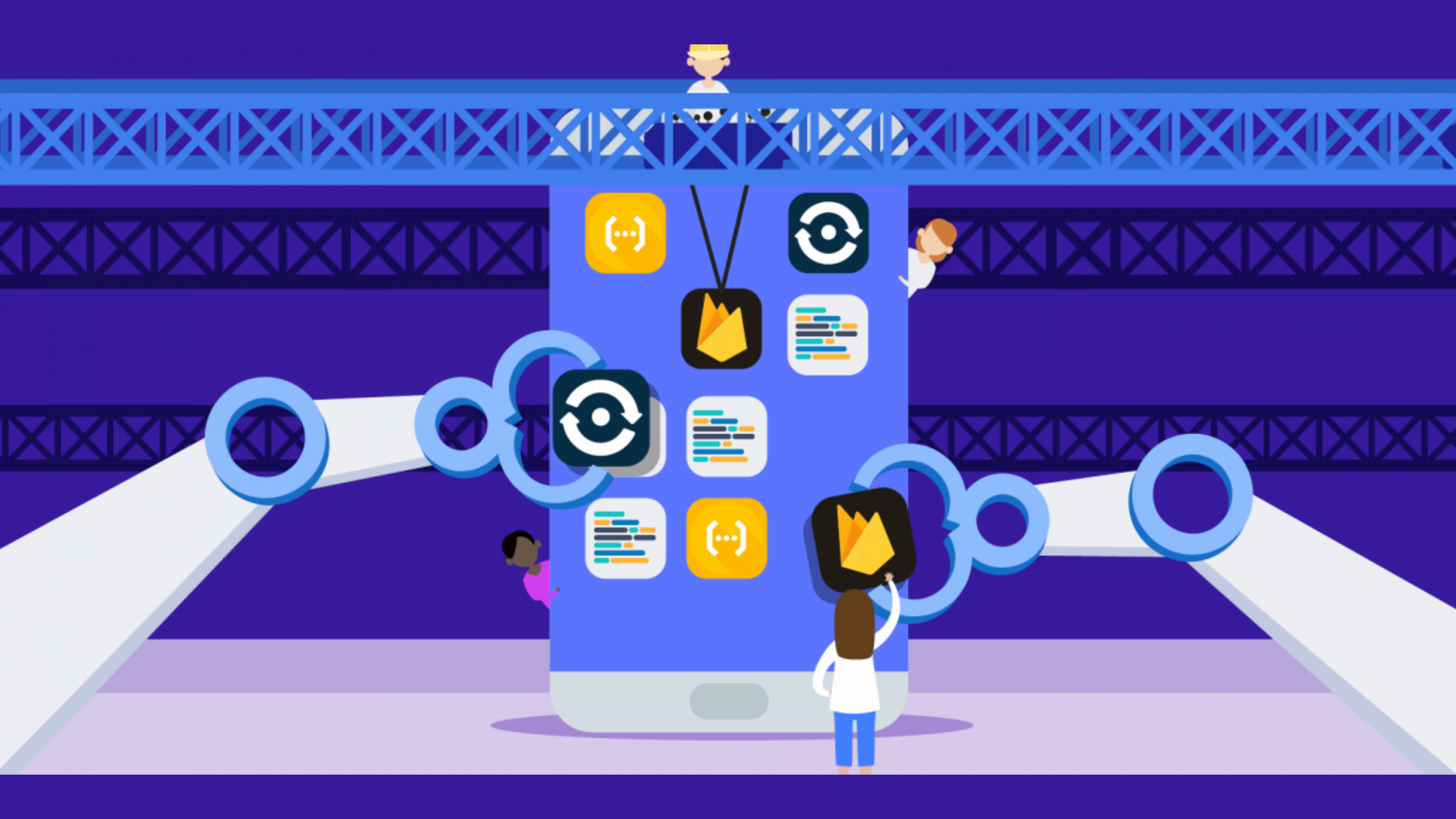


nexmo® | The Vonage® API Platform

Add Two Factor Authentication to a Microsoft Bot

with Nexmo's Verify API







ADD TWO FACTOR AUTHENTICATION

to iOS Apps with Swift and Nexmo's Verify API

iOS

(No Model.) 2 Sheets—Sheet 1.

C. P. WEST.

No. 263,397. Patented Aug. 29, 1882.



Fig. 1.

WITNESSES:
H. Raffloub
Otto Rasch

INVENTOR
Charles P. West
BY Munn & Co.
ATTORNEY

COPY






It was
4am
actually

TASK - SERVE THIS JSON RESPONSE

```
[
    {
        "action": "talk",
        "text": "G'day PyCon Australia, this is Nexmo"
    }
]
```



Flask

web development,
one drop at a time

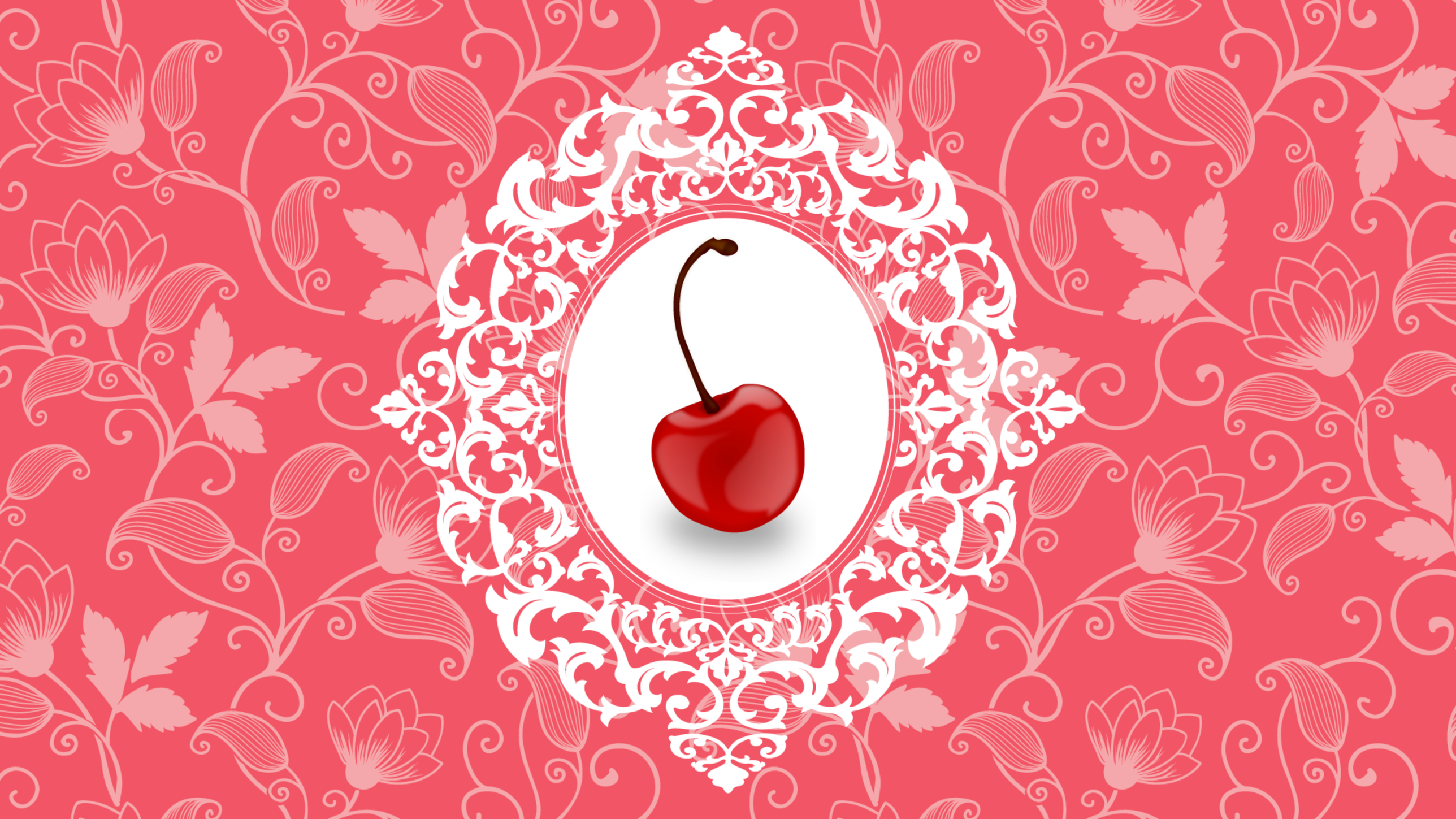
```python
from flask import Flask, jsonify

app = Flask(__name__)


@app.route("/ncco")
def ncco():
    return jsonify([{"action": "talk", "text": "Hello World from Flask"}])
```

```
export FLASK_APP=hello.py
flask run
```

flask.pocoo.org



```python
import cherrypy


class NCCOView(object):
    @cherrypy.expose
    @cherrypy.tools.json_out()
    def index(self):
        return [{"action": "talk", "text": "Hello World from CherryPy"}]


if __name__ == "__main__":
    cherrypy.quickstart(NCCOView(), "/ncco")
```

cherrypy.org



```python
import falcon


class NCCOView:
    def on_get(self, req, resp):
        """Handles GET requests"""
        ncco = {"action": "talk", "text": "Hello World from Falcon"}

        resp.media = ncco


api = falcon.API()
api.add_route("/ncco", NCCOView())
```

falconframework.org

```python
import hug


@hug.get("/ncco")
def ncco():
    return [{"action": "talk", "text": "Hello World from Hug"}]
```

```python
import falcon


class NCCOView:
    def on_get(self, req, resp):
        """Handles GET requests"""
        ncco = {"action": "talk", "text": "Hello World from Falcon"}

        resp.media = ncco


api = falcon.API()
api.add_route("/ncco", NCCOView())
```

```python
import hug


@hug.get("/ncco")
def ncco():
    return [{"action": "talk", "text": "Hello World from Hug"}]
```

```python
import hug


@hug.get("/ncco", versions=1)
def ncco():
    return [{"action": "talk", "text": "Hello World from Hug"}]


@hug.get("/ncco", examples="who=World", versions=range(2, 5))
def ncco(who: hug.types.text):
    return [{"action": "talk", "text": f"Hello {who} from Hug"}]
```

```
{
    "404": "The API call you tried to make was not defined.",
    "documentation": {
        "version": 4,
        "versions": [
            1,
            2,
            3,
            4
        ],
        "handlers": {
            "/ncco": {
                "GET": {
                    "examples": [
                        "http://127.0.0.1:8000/v4/ncco?who=World"
                    ],
                    "outputs": {
                        "format": "JSON (Javascript Serialized Object Notation)",
                        "content_type": "application/json; charset=utf-8"
                    },
                    "inputs": {
                        "who": {
                            "type": "Basic text / string value"
                        }
                    }
                }
            }
        }
    }
}
```


www.hug.rest

```python
from flask import Flask, jsonify

app = Flask(__name__)


@app.route("/ncco")
def ncco():
    return jsonify([{"action": "talk", "text": "Hello World from Flask"}])
```

```python
import cherrypy


class NCCOView(object):
    @cherrypy.expose
    @cherrypy.tools.json_out()
    def index(self):
        return [{"action": "talk", "text": "Hello World from CherryPy"}]


if __name__ == "__main__":
    cherrypy.quickstart(NCCOView(), "/ncco")
```

```python
import falcon


class NCCOView:
    def on_get(self, req, resp):
        """Handles GET requests"""
        ncco = {"action": "talk", "text": "Hello World from Falcon"}

        resp.media = ncco


api = falcon.API()
api.add_route("/ncco", NCCOView())
```

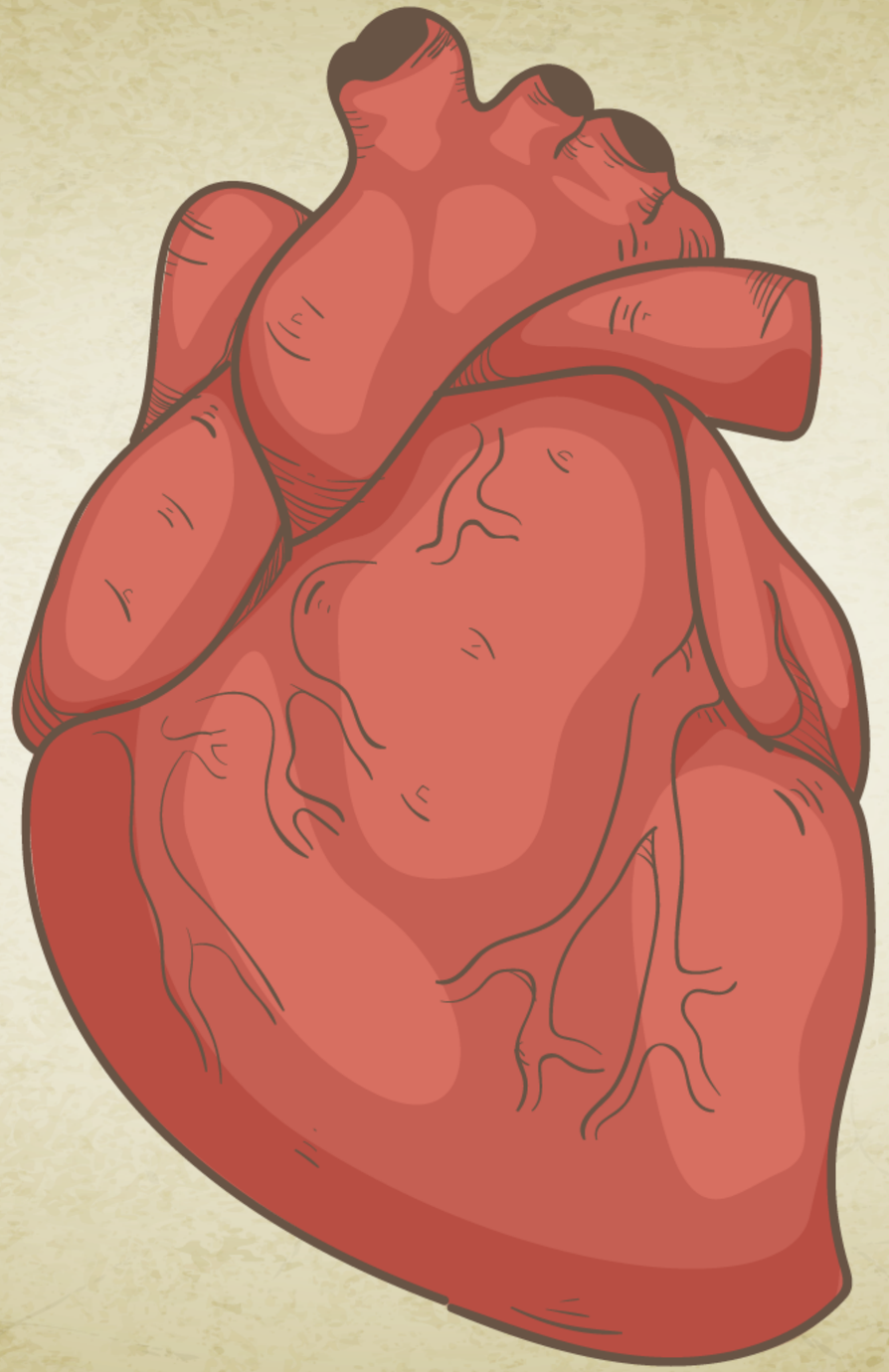
```python
import hug

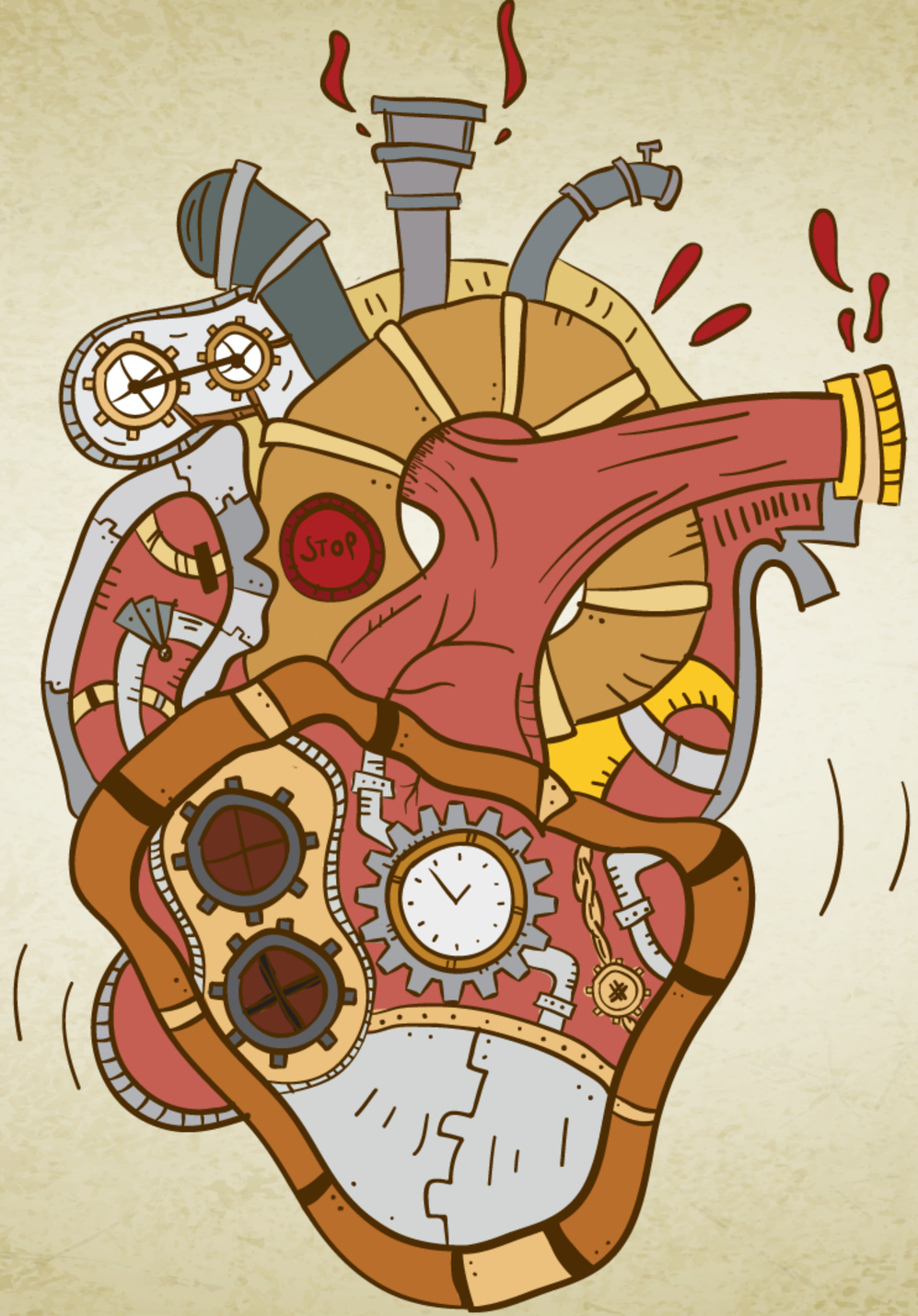

@hug.get("/ncco")
def ncco():
    return [{"action": "talk", "text": "Hello World from Hug"}]
```


Freedom







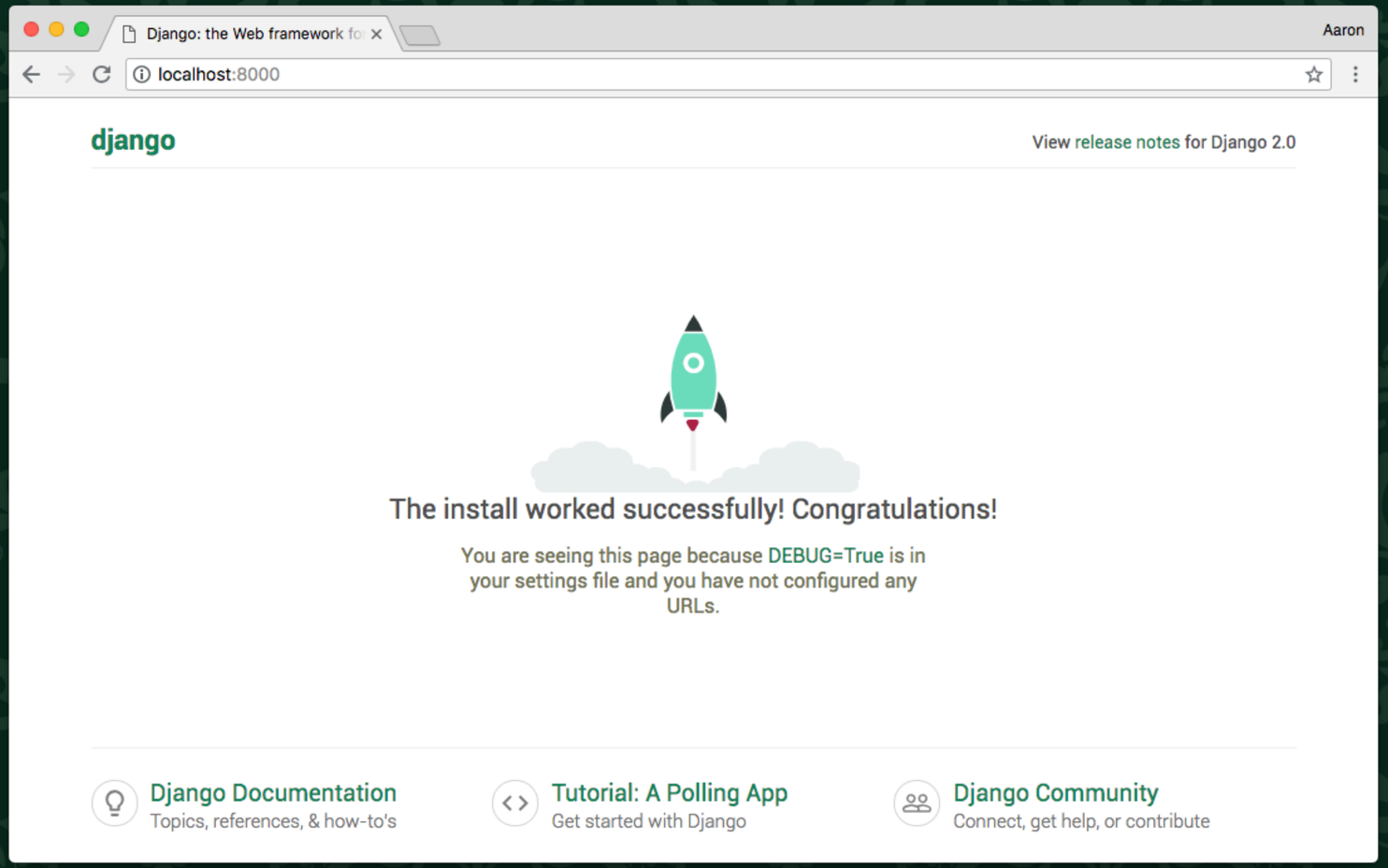

STOP

Batteries Included


django

```
django-admin startproject hello_django
```

```
python manage.py runserver
```



```
python manage.py startapp conversation
```

```
1. aaronbassett@Aarons-MBP: ~/Projects/hello-world-x8/hello_django (zsh)

 hello-world-x8-jSO2sze5  ...Projects/hello-world-x8/hello_django
 tree conversation
conversation
├── __init__.py
├── admin.py
├── apps.py
├── migrations
│   └── __init__.py
├── models.py
├── tests.py
└── views.py

1 directory, 7 files

 hello-world-x8-jSO2sze5  ...Projects/hello-world-x8/hello_django
```



```python
# settings.py

INSTALLED_APPS = [
    ...
    "conversation.apps.ConversationConfig",
]
```

```python
# urls.py

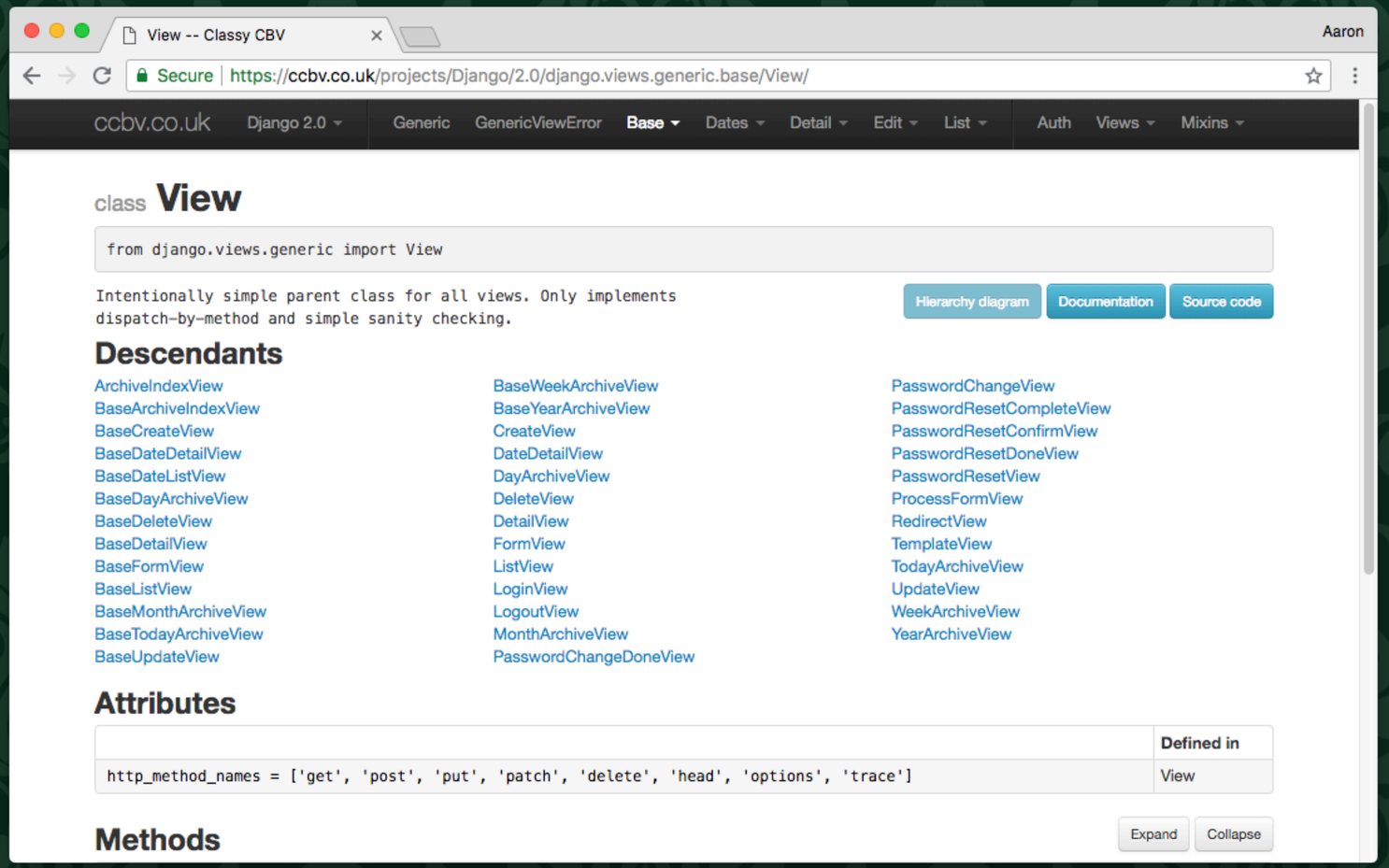
from django.urls import path, include

urlpatterns = [path("ncco/", include("conversation.urls"))]
```

```python
# views.py

from django.views.generic import View
from django.http import JsonResponse


class NCCOView(View):
    def get(self, request, *args, **kwargs):
        return JsonResponse(
            [{"action": "talk", "text": "Hello World from Django"}], safe=False
        )
```


View -- Classy CBV
Aaron
Secure | https://ccbv.co.uk/projects/Django/2.0/django.views.generic.base/View/
ccbv.co.uk
Django 2.0
Generic
GenericViewError
Base
Dates
Detail
Edit
List
Auth
Views
Mixins
class View
from django.views.generic import View
Intentionally simple parent class for all views. Only implements dispatch-by-method and simple sanity checking.
Hierarchy diagram
Documentation
Source code
Descendants
ArchiveIndexView
BaseArchiveIndexView
BaseCreateView
BaseDateDetailView
BaseDateListView
BaseDayArchiveView
BaseDeleteView
BaseDetailView
BaseFormView
BaseListView
BaseMonthArchiveView
BaseTodayArchiveView
BaseUpdateView
BaseWeekArchiveView
BaseYearArchiveView
CreateView
DateDetailView
DayArchiveView
DeleteView
DetailView
FormView
ListView
LoginView
LogoutView
MonthArchiveView
PasswordChangeDoneView
PasswordChangeView
PasswordResetCompleteView
PasswordResetConfirmView
PasswordResetDoneView
PasswordResetView
ProcessFormView
RedirectView
TemplateView
TodayArchiveView
UpdateView
WeekArchiveView
YearArchiveView
Attributes
Defined in
http_method_names = ['get', 'post', 'put', 'patch', 'delete', 'head', 'options', 'trace']
View
Methods
Expand
Collapse

ccbv.co.uk 2

```python
# views.py

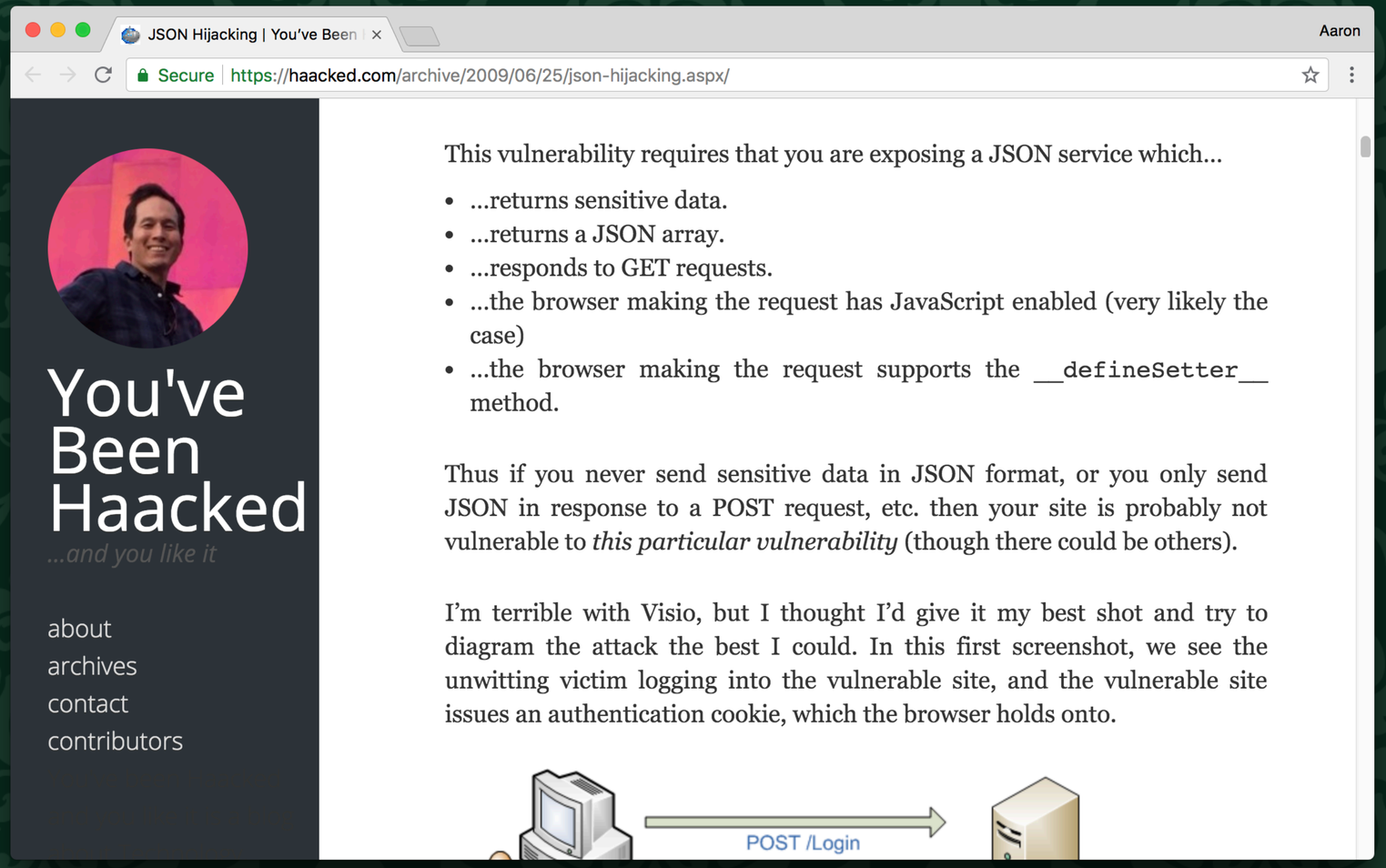
from django.views.generic import View
from django.http import JsonResponse


class NCCOView(View):
    def get(self, request, *args, **kwargs):
        return JsonResponse(
            [{"action": "talk", "text": "Hello World from Django"}], safe=False
        )
```

```python
# views.py

from django.views.generic import View
from django.http import JsonResponse


class NCCOView(View):
    def get(self, request, *args, **kwargs):
        return JsonResponse(
            [{"action": "talk", "text": "Hello World from Django"}], safe=False
        )
```



haacked.com/archive/2009/06/25/json-hijacking.aspx/

```python
# urls.py

from django.conf.urls import url
from conversation.views import NCCOView

app_name = "conversation"

urlpatterns = [url(r"^$", NCCOView.as_view(), name="ncco")]
```

```python
# settings.py

INSTALLED_APPS = [
    ...
    "conversation.apps.ConversationConfig",
]


# urls.py

from django.urls import path, include

urlpatterns = [path("ncco/", include("conversation.urls"))]


# conversation/views.py

from django.views.generic import View
from django.http import JsonResponse


class NCCOView(View):
    def get(self, request, *args, **kwargs):
        return JsonResponse(
            [{"action": "talk", "text": "Hello World from Django"}], safe=False
        )


# conversation/urls.py

from django.conf.urls import url
from conversation.views import NCCOView

app_name = "conversation"

urlpatterns = [url(r"^$", NCCOView.as_view(), name="ncco")]
```

www.djangoproject.com


PYRAMID



START SMALL



FINISH BIG

```python
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.view import view_config


@view_config(renderer="json")
def ncco(request):
    return [{"action": "talk", "text": "Hello World from Pyramid"}]


if __name__ == "__main__":
    with Configurator() as config:
        config.add_route("ncco", "/ncco")
        config.add_view(ncco, route_name="ncco", renderer="json")
        app = config.make_wsgi_app()

    server = make_server("0.0.0.0", 8000, app)
    server.serve_forever()
```

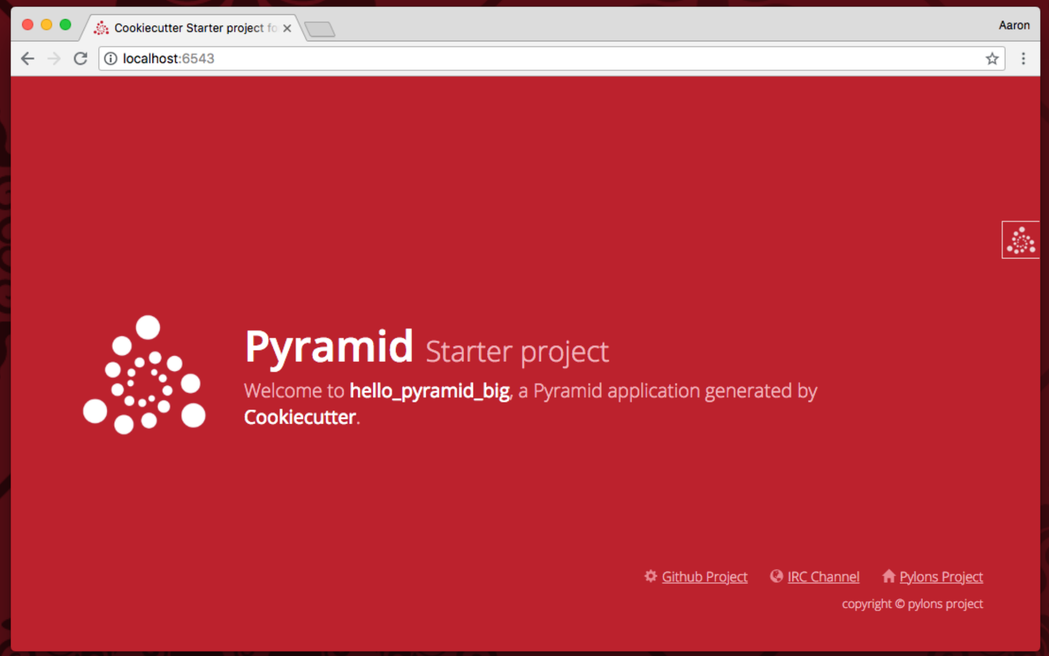


FINISH BIG

```
cookiecutter gh:Pylons/pyramid-cookiecutter-starter
```

```
pipenv install -e hello_pyramid_big
```

```
./env/bin/pserve development.ini
```



```python
# init.py

from pyramid.config import Configurator


def main(global_config, **settings):
    config = Configurator(settings=settings)
    config.add_route("ncco", "/ncco")
    config.scan(".views")
    return config.make_wsgi_app()
```

```python
# views.py

from pyramid.view import view_config, view_defaults


@view_defaults(renderer="json")
class NCCOViews:
    def __init__(self, request):
        self.request = request

    @view_config(route_name="ncco")
    def ncco(self):
        return [{"action": "talk", "text": "Hello World from Pyramid"}]
```

trypyramid.com

```python
# settings.py

INSTALLED_APPS = [
    ...
    "conversation.apps.ConversationConfig",
]


# urls.py

from django.urls import path, include

urlpatterns = [path("ncco/", include("conversation.urls"))]


# conversation/views.py

from django.views.generic import View
from django.http import JsonResponse


class NCCOView(View):
    def get(self, request, *args, **kwargs):
        return JsonResponse(
            [{"action": "talk", "text": "Hello World from Django"}], safe=False
        )


# conversation/urls.py

from django.conf.urls import url
from conversation.views import NCCOView

app_name = "conversation"

urlpatterns = [url(r"^$", NCCOView.as_view(), name="ncco")]
```

```python
# init.py

from pyramid.config import Configurator


def main(global_config, **settings):
    config = Configurator(settings=settings)
    config.add_route("ncco", "/ncco")
    config.scan(".views")
    return config.make_wsgi_app()


# views.py

from pyramid.view import view_config, view_defaults


@view_defaults(renderer="json")
class NCCOViews:
    def __init__(self, request):
        self.request = request

    @view_config(route_name="ncco")
    def ncco(self):
        return [{"action": "talk", "text": "Hello World from Pyramid"}]
```

Tornado

```python
import tornado.ioloop
import tornado.web


class NCCOHandler(tornado.web.RequestHandler):
    def get(self):
        self.write({"action": "talk", "text": "Hello World from Tornado"})


def make_app():
    return tornado.web.Application([(r"/ncco", NCCOHandler)])

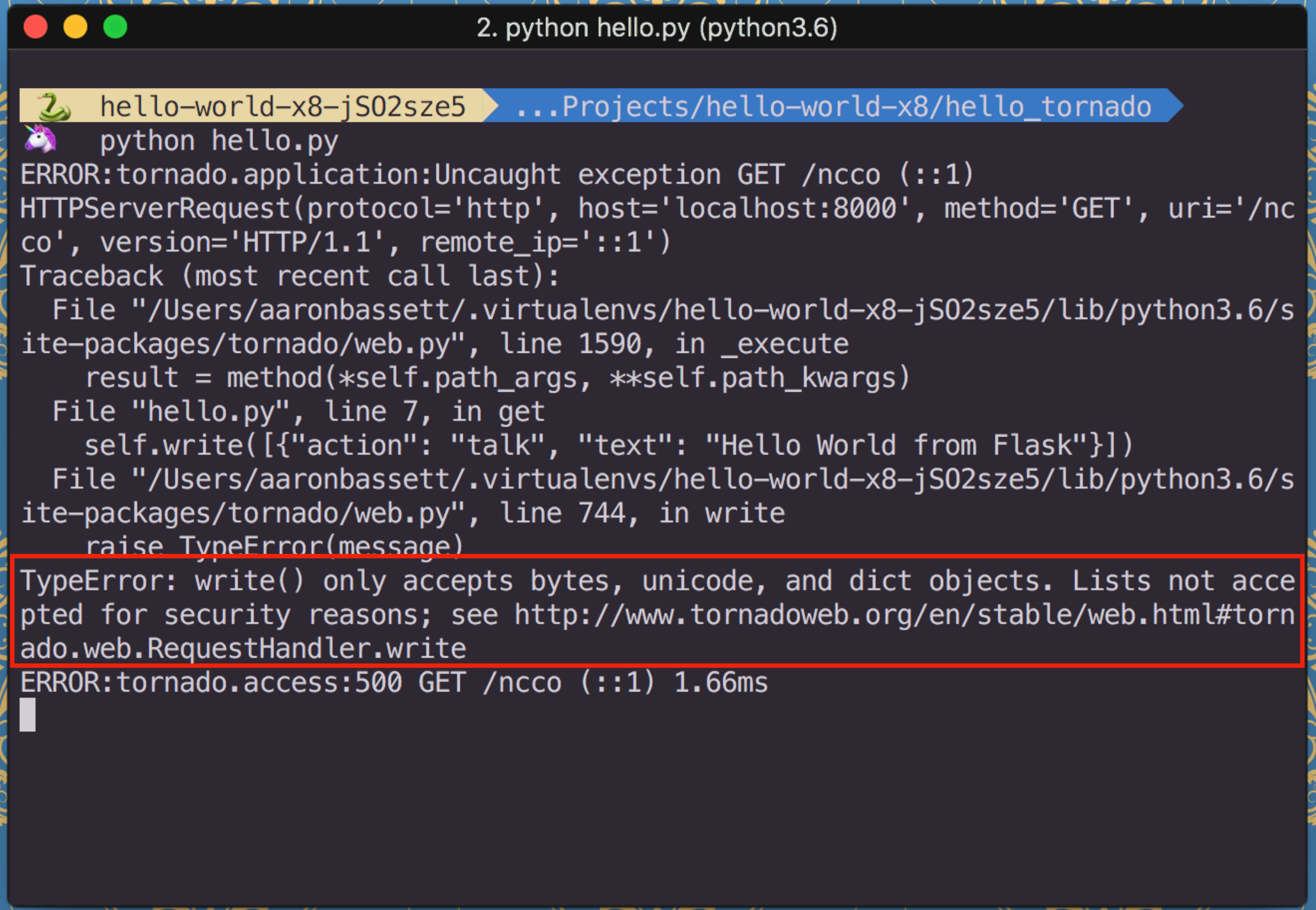

if __name__ == "__main__":
    app = make_app()
    app.listen(8000)
    tornado.ioloop.IOLoop.current().start()
```

```python
import tornado.ioloop
import tornado.web


class NCCOHandler(tornado.web.RequestHandler):
    def get(self):
        self.write({"action": "talk", "text": "Hello World from Tornado"})


def make_app():
    return tornado.web.Application([(r"/ncco", NCCOHandler)])


if __name__ == "__main__":
    app = make_app()
    app.listen(8000)
    tornado.ioloop.IOLoop.current().start()
```


2. python hello.py (python3.6)
hello-world-x8-jSO2sze5
...Projects/hello-world-x8/hello_tornado
python hello.py
ERROR:tornado.application:Uncaught exception GET /ncco (::1)
HTTPServerRequest(protocol='http', host='localhost:8000', method='GET', uri='/ncco', version='HTTP/1.1', remote_ip='::1')
Traceback (most recent call last):
File "/Users/aaronbassett/.virtualenvs/hello-world-x8-jSO2sze5/lib/python3.6/site-packages/tornado/web.py", line 1590, in _execute
result = method(*self.path_args, **self.path_kwargs)
File "hello.py", line 7, in get
self.write([{"action": "talk", "text": "Hello World from Flask"}])
File "/Users/aaronbassett/.virtualenvs/hello-world-x8-jSO2sze5/lib/python3.6/site-packages/tornado/web.py", line 744, in write
raise TypeError(message)
TypeError: write() only accepts bytes, unicode, and dict objects. Lists not accepted for security reasons; see http://www.tornadoweb.org/en/stable/web.html#tornado.web.RequestHandler.write
ERROR:tornado.access:500 GET /ncco (::1) 1.66ms



I WILL FIND A WAY TO SHOOT MYSELF IN THE FOOT

```
def write(self, chunk):
    if self._finished:
        raise RuntimeError("Cannot write() after finish()")
    if not isinstance(chunk, (bytes, unicode_type, dict)):
        message = "write() only accepts bytes, unicode, and dict objects"
        if isinstance(chunk, list):
            message += (
                ". Lists not accepted for security reasons; see "
                + "http://www.tornadoweb.org/en/stable/web.html#tornado.web.RequestHandler.write"
            )
        raise TypeError(message)
    if isinstance(chunk, dict):
        chunk = escape.json_encode(chunk)
        self.set_header("Content-Type", "application/json; charset=UTF-8")
    chunk = utf8(chunk)
    self._write_buffer.append(chunk)
```

```python
def write(self, chunk):
    if self._finished:
        raise RuntimeError("Cannot write() after finish()")
    if not isinstance(chunk, (bytes, unicode_type, dict)):
        message = "write() only accepts bytes, unicode, and dict objects"
        if isinstance(chunk, list):
            message += (
                ". Lists not accepted for security reasons; see "
                + "http://www.tornadoweb.org/en/stable/web.html#tornado.web.RequestHandler.write"
            )
        raise TypeError(message)
    if isinstance(chunk, dict):
        chunk = escape.json_encode(chunk)
        self.set_header("Content-Type", "application/json; charset=UTF-8")
    chunk = utf8(chunk)
    self._write_buffer.append(chunk)
```

```python
import tornado.ioloop
import tornado.web
from tornado import escape
from tornado.escape import utf8


class NCCOHandler(tornado.web.RequestHandler):
    def write(self, chunk):
        chunk = escape.json_encode(chunk)
        self.set_header("Content-Type", "application/json; charset=UTF-8")
        chunk = utf8(chunk)
        self._write_buffer.append(chunk)

    def get(self):
        self.write([{"action": "talk", "text": "Hello World from Tornado"}])


def make_app():
    return tornado.web.Application([(r"/ncco", NCCOHandler)])


if __name__ == "__main__":
    app = make_app()
    app.listen(8000)
    tornado.ioloop.IOLoop.current().start()
```

www.tornadoweb.org


Sanic
gotta go fast

GOTTA GO



FAST

2. python hello.py (python3.6)

```
🦄  python hello.py
[2018-07-25 17:41:37 +0100] [33459] [DEBUG]
```




```
[2018-07-25 17:41:37 +0100] [33459] [INFO] Goin' Fast @ http://0.0.0.0:8000
[2018-07-25 17:41:37 +0100] [33459] [INFO] Starting worker [33459]
```

```python
from sanic import Sanic
from sanic.response import json

app = Sanic()


@app.route("/ncco")
async def ncco(request):
    return json([{"action": "talk", "text": "Hello World from Sanic"}])


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

```python
from sanic import Sanic
from sanic.views import HTTPMethodView
from sanic.response import json

app = Sanic("nexmo_app")


class NCCOAsyncView(HTTPMethodView):
    async def get(self, request):
        return json([{"action": "talk", "text": "Hello World from Sanic"}])


app.add_route(NCCOAsyncView.as_view(), "/ncco")


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```


anic
gotta DIE fast

sanic.readthedocs.io

```python
from flask import Flask, jsonify

app = Flask(__name__)


@app.route("/ncco")
def ncco():
    return jsonify([{"action": "talk", "text": "Hello World from Flask"}])
```

```python
from sanic import Sanic
from sanic.response import json

app = Sanic()


@app.route("/ncco")
async def ncco(request):
    return json([{"action": "talk", "text": "Hello World from Sanic"}])


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

```python
import hug


@hug.get("/ncco")
def ncco():
    return [{"action": "talk", "text": "Hello World from Hug"}]
```

```python
# views.py

from pyramid.view import view_config, view_defaults


@view_defaults(renderer="json")
class NCCOViews:
    def __init__(self, request):
        self.request = request

    @view_config(route_name="ncco")
    def ncco(self):
        return [{"action": "talk", "text": "Hello World from Pyramid"}]
```

```python
from sanic import Sanic
from sanic.views import HTTPMethodView
from sanic.response import json

app = Sanic("nexmo_app")


class NCCOAsyncView(HTTPMethodView):
    async def get(self, request):
        return json([{"action": "talk", "text": "Hello World from Sanic"}])


app.add_route(NCCOAsyncView.as_view(), "/ncco")


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

```python
import tornado.ioloop
import tornado.web
from tornado import escape
from tornado.escape import utf8


class NCCOHandler(tornado.web.RequestHandler):
    def write(self, chunk):
        chunk = escape.json_encode(chunk)
        self.set_header("Content-Type", "application/json; charset=UTF-8")
        chunk = utf8(chunk)
        self._write_buffer.append(chunk)

    def get(self):
        self.write([{"action": "talk", "text": "Hello World from Tornado"}])


def make_app():
    return tornado.web.Application([(r"/ncco", NCCOHandler)])


if __name__ == "__main__":
    app = make_app()
    app.listen(8000)
    tornado.ioloop.IOLoop.current().start()
```

```python
# views.py

from django.views.generic import View
from django.http import JsonResponse


class NCCOView(View):
    def get(self, request, *args, **kwargs):
        return JsonResponse(
            [{"action": "talk", "text": "Hello World from Django"}], safe=False
        )
```

```python
import falcon


class NCCOView:
    def on_get(self, req, resp):
        """Handles GET requests"""
        ncco = {"action": "talk", "text": "Hello World from Falcon"}

        resp.media = ncco


api = falcon.API()
api.add_route("/ncco", NCCOView())
```

```python
import falcon


class NCCOView:
    def on_get(self, req, resp):
        """Handles GET requests"""
        ncco = {"action": "talk", "text": "Hello World from Falcon"}

        resp.media = ncco


api = falcon.API()
api.add_route("/ncco", NCCOView())
```

```python
import tornado.ioloop
import tornado.web
from tornado import escape
from tornado.escape import utf8


class NCCOHandler(tornado.web.RequestHandler):
    def write(self, chunk):
        chunk = escape.json_encode(chunk)
        self.set_header("Content-Type", "application/json; charset=UTF-8")
        chunk = utf8(chunk)
        self._write_buffer.append(chunk)

    def get(self):
        self.write([{"action": "talk", "text": "Hello World from Tornado"}])

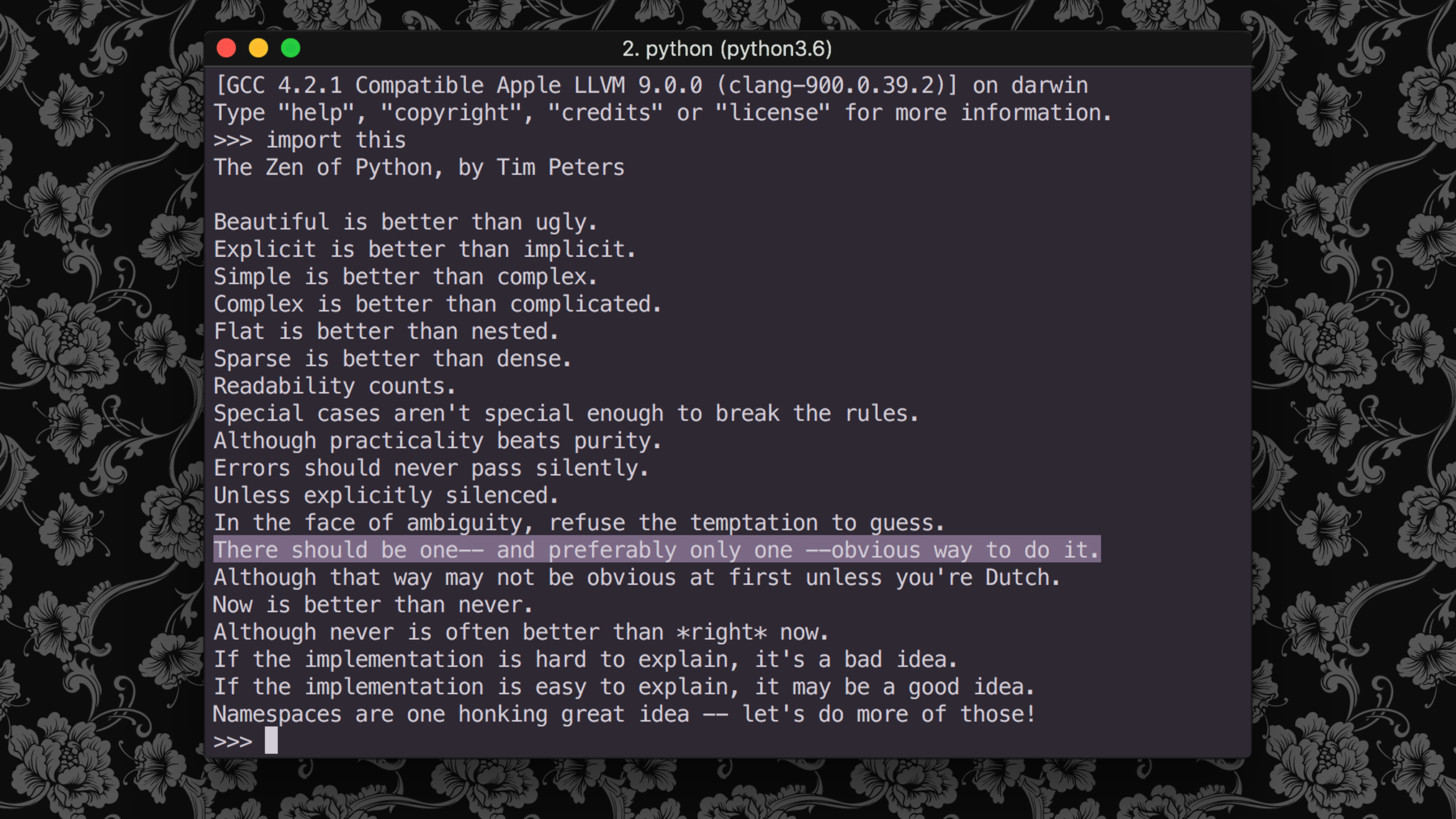

def make_app():
    return tornado.web.Application([(r"/ncco", NCCOHandler)])


if __name__ == "__main__":
    app = make_app()
    app.listen(8000)
    tornado.ioloop.IOLoop.current().start()
```

```python
# init.py

from pyramid.config import Configurator


def main(global_config, **settings):
    config = Configurator(settings=settings)
    config.add_route("ncco", "/ncco")
    config.scan(".views")
    return config.make_wsgi_app()
```


2. python (python3.6)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>

- Documentation
- Bug / Issue tracker
- Release management / Deprecation process
- Community


@AARONBASSETT