



Python. И веб-разработка.



Веб-разработка



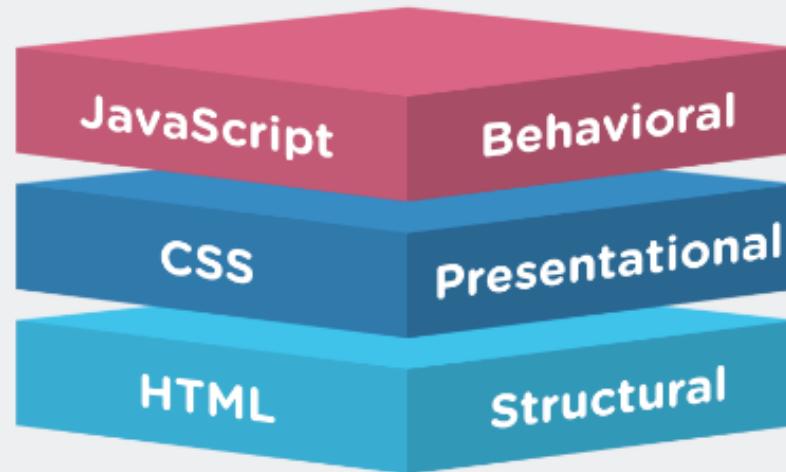
Что это?

Составные части



Front end vs. Back end.

Фронтенд





HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>My title</title>
    </head>
    <body>
        <h1>Heading</h1>
        <p>Paragraph.</p>
    </body>
</html>
```

CSS



```
h1 {  
    color: blue;  
    font-family: verdana;  
    font-size: 300%;  
}  
p {  
    color: red;  
    font-family: courier;  
    font-size: 160%;  
}
```



JavaScript

```
function myFunction() {  
    var x = document.getElementById("form");  
    var text = "";  
    var i;  
    for (i = 0; i < x.length; i++) {  
        text += x.elements[i].value + "<br>";  
    }  
    document.getElementById("output").innerHTML = text;  
}
```

Подходы к организации web-приложений



- Отдельные ресурсы для каждого адреса
- Single-page applications (SPA)



Популярные js- библиотеки

- Angular JS
- ReactJS
- Миллион других

Темная сторона бекенда



Почему Python?

Темная сторона бекенда



"Чистый" Python или
фреймворк?



Что должен уметь бекенд

- Бизнес-логика приложения, работа с базой данных
- Входные точки - обработка HTTP запросов
- Подготовка ответа - сериализованные данные (JSON) или server-side шаблонизация
- Авторизация
- Кеширование, логирование, отказоустойчивость, производительность

Python web-фреймворки

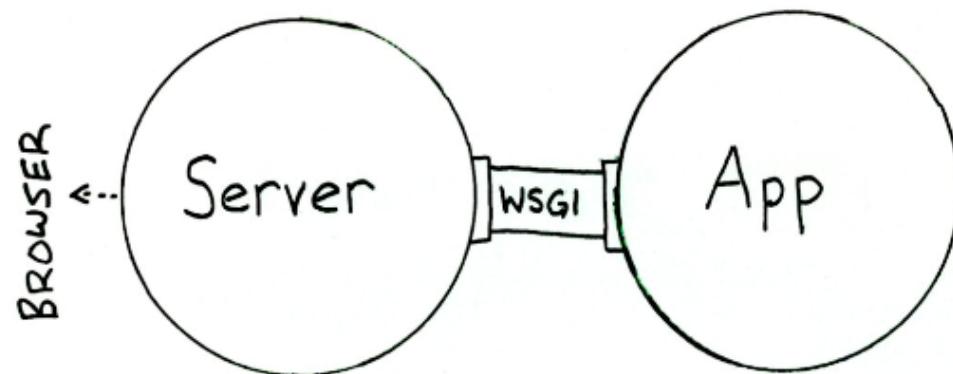


- Django
- Flask
- Bottle
- Tornado
- Многие другие

WSGI



<https://www.python.org/dev/peps/pep-0333/>



Bottle



```
from bottle import route, run, template

@route('/hello/<name>')
def index(name):
    return template('<b>Hello {{name}}!</b>', name=name)

run(host='localhost', port=8080)
```

Flask



```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Tornado



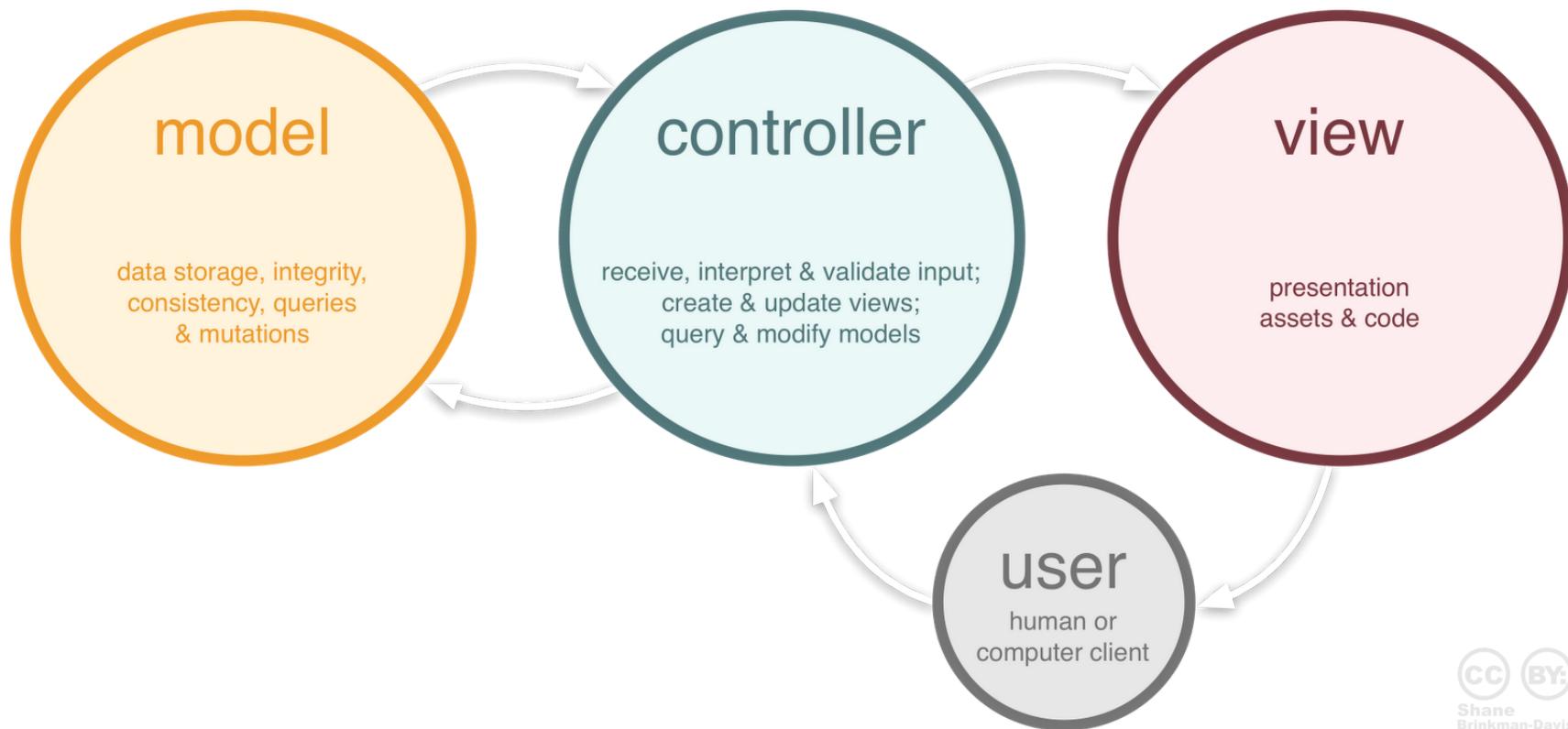
```
import tornado.ioloop
import tornado.web

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")

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

if __name__ == "__main__":
    app = make_app()
    app.listen(8888)
    tornado.ioloop.IOLoop.current().start()
```

MVC



Django



- ORM, абстракция над несколькими RDBMS
- Аутентификация, сессии
- Авторизация
- Роутинг
- Шаблоны
- Миграции
- Админка
- Формы
- Интернационализация и локализация
- Кеширование
- Многое другое

WSGI-совместимые веб-серверы



- Gunicorn
- uWSGI
- Chaussette



Python и real-time web



REAL-TIME СООБЩЕНИЯ

Сообщение о событии,
доставленное клиенту
в кратчайшие сроки
после возникновения



**200 МИЛЛИСЕКУНД
ИЛИ БАЯН!**



Зачем?



Krypt 16 апреля 2011 в 12:29 # ★ 5 1

+2

Я всегда буду обновлять страницу перед написанием комментариев.

Я всегда буду обновлять страницу перед написанием комментариев.

[ответить](#)

ВАУ-ЭФФЕКТ

Total	
Tweet	
18176	
Words	
178222	
Chars	
1085745	
Last #hashtag	
#amazed	
Last @mentions	
@CarlKennedy96	
@chloebennettxo	

Tweetping

Check out the Twitter activity in realtime



[J'aime](#) 14k [Envoyer](#)

[Tweet](#) 13.1K

This project is
based on Nodejs w/
Socket.io,
Processing.js and
Backbone.js

Designed and
realized by Franck
Frenzenin

north america

Tweet	
5773	
Words	
58074	

[Last #hashtag](#)

#toomuchtoaskfor

[Last @mentions](#)

@_wattts

south america

Tweet	
2686	
Words	
24483	

[Last #hashtag](#)

#Llamadas2013

[Last @mentions](#)

@valenstaffora

europe

Tweet	
7607	
Words	
74707	

[Last #hashtag](#)

#amazed

[Last @mentions](#)

@CarlKennedy96

@chloebennettxo

africa

Tweet	
739	
Words	
8665	

[Last #hashtag](#)

#0fMonstersAndMan.

[Last @mentions](#)

@BlackouT_MW

asia

Tweet	
1287	
Words	
11396	

[Last #hashtag](#)

#IDon'tLike

[Last @mentions](#)

@takafumiakio000:



Как доставить контент моментально?

- Long-polling
- XHR-streaming
- Eventsource (SSE)
- Websockets

НЕЛЬЗЯ ПРОСТО ТАК ВЗЯТЬ



**И НАЧАТЬ ИСПОЛЬЗОВАТЬ
WEBSOCKETS**

Библиотеки-полифиллы



Socket.io



SockJS



Если на бекенде Python

- Асинхронный фреймворк (Tornado, Asyncio)
- Gevent - патчим стандартную библиотеку
- Nginx-push-stream module
- Hosted-сервер с API (LightStreamer, Centrifugo)
- Облачный сервис с API (pusher.com, pubnub.com)
- Django-channels

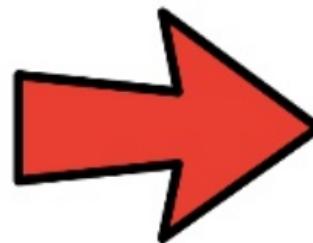


Centrifugo

Real-time messaging

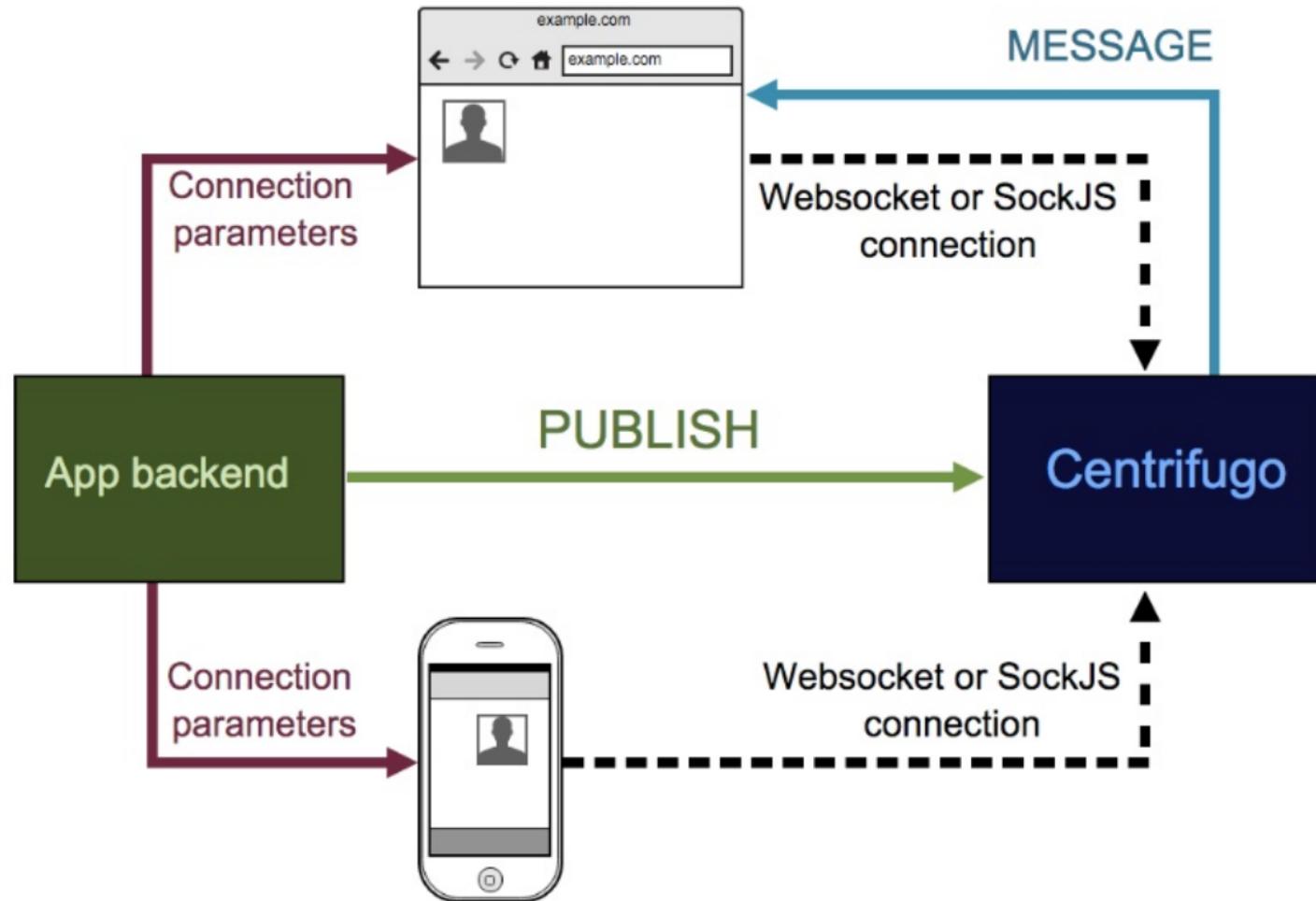


Python
Centrifuge



Go
Centrifugo

Схема работы





Использование в браузере

```
var centrifuge = new Centrifuge({  
    url: "http://centrifugo.example.com",  
    user: "2694",  
    timestamp: "1447969543",  
    token: "HMAC SHA-256 token"  
});  
  
centrifuge.subscribe("news", function(message) {  
    console.log(message);  
});  
  
centrifuge.connect();
```



Особенности

- Информация о подключениях в канале
- События подписки на канал/отписки от канала
- История сообщений в канале
- Восстановление пропущенных сообщений
- Разные типы каналов
- Масштабирование с помощью Redis
- Метрики
- Административный веб-интерфейс
- RPM и DEB пакеты, Docker-контейнер

API и клиентские библиотеки



- Клиентские - браузер, Android, iOS
- HTTP API - Python, PHP, Go, NodeJS, Ruby

Ссылки



- Код - <https://github.com/centrifugal/centrifugo>
- [Документация](#)
- [Демо-инстанс](#) на Heroku (пароль demo)



Спасибо!