Nodejs直出套路

——一个边际开发成本最小化的直出框架

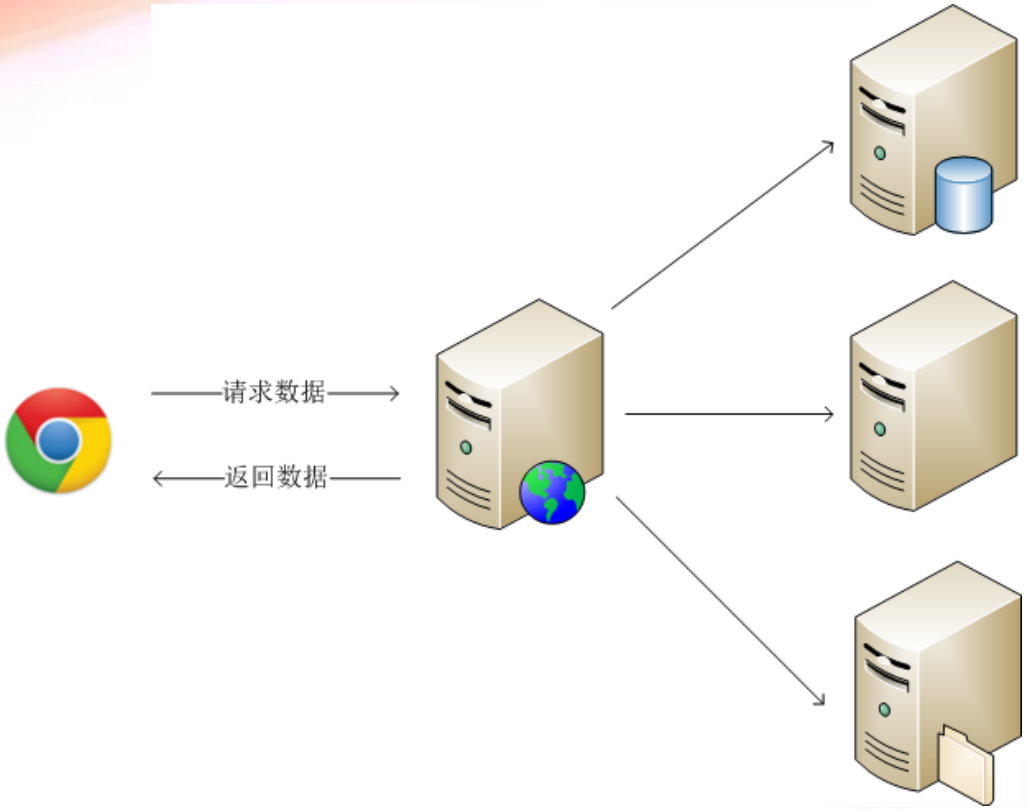
腾讯 姚穗斌

腾讯视频 V.QQ.COM

2016年11月

Web server的特点是什么

Web服务器一般做什么



1. 接收请求
2. 从不同的后台通过网络获取所需数据
3. 对数据进行加工处理
4. 返回结果

Web服务器特点：I/O操作多，计算简单

定一个小目标，用Nodejs做直出页面

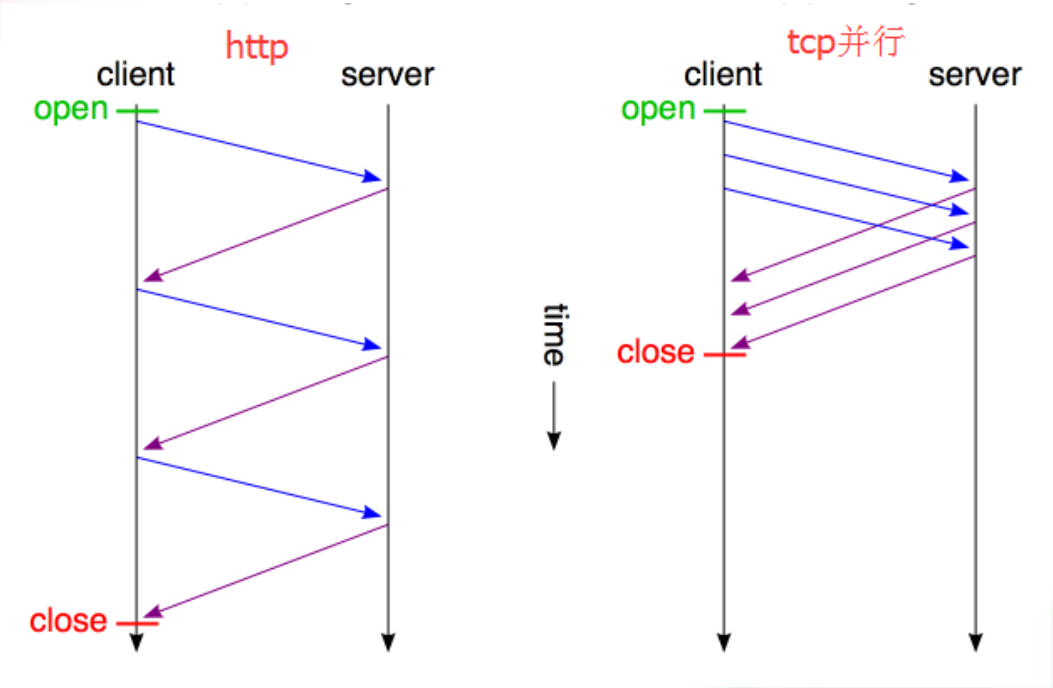
- Step 1，先打通各种后台server
  - Http(s)
  - protobuf协议
  - Redis
  - Mysql
  - 私有协议
  - UDP协议
  - …

- 外部协议都能找到npm包了，但私有协议怎么搞？

TCP Socket请求的好处

节省连接数，一个TCP连接可满足多并发请求

http请求下，1000个并发就需要创建1000个TCP连接



Http短连接 vs TCP长连接对比试验

请求A：内网http接口
请求B：用protobuf协议的server
两个服务的逻辑相同，都是从Redis获取数据然后返回

请求方式：各发起1000个并发请求，每个请求的回包大小为2k

```
Http request: 1271ms
Http request ended. Success 1000, Fail 0
Protobuf request: 581ms
Protobuf request ended. Success 1000, Fail 0
```

结论：
Http最大请求数 780/S
Protobuf最大请求数 1720/S
二进制长连接比Http短连接性能提升200%+

套路一：TCP Socket请求

1. 建立TCP连接，发送请求，将json转换成Buffer流

```
var seq = 0;
var data = {
    id : 123,
    from : "web"
};

var buf = encode(data, ++seq);
//buf <Buffer 10 d1 0f 1a 0b ..
socket.write(buf);
```

2. 接收请求，将二进制格式的Buffer转换为json对象

```
socket.on('data', function(data) {
    var buf;
    //do something..

    var obj = decode(buf); //<Buffer 10 d1 0f 1a 0b ..
    //obj {id:123, name:"victor"}
}
```

3. 判断数据包是否完整
    a. 未接收完的buf
    b. 包含多个包的buf

套路一： TCP Socket请求

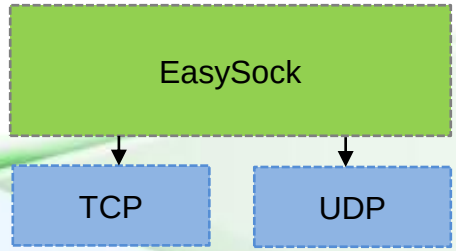
```
var net = require('net');
var client = new net.Socket();

var encode = function(){...};
var decode = function(){...};
var check = function(){...};

client.connect('127.0.0.1', 1234, function() {
    var data = [1,2,3];
    client.write(encode(data));
});

client.on('data', function(buf) {
    if(check(buf)){
      var json = decode(buf);
      console.log(json);
    }
    //...
});
```

- 大部分网络协议，差别仅仅在于这三个方法，其他过程都一样
  - encode() 封包
  - decode() 解包
  - check() 判断数据包是否完整

EasySock模块

- 为解决以下问题而生：
  - 处理socket发送、接收数据，以及数据包完整性校验
  - 封装网络请求中各种复杂的异步调用以及中间过程，屏蔽tcp连接细节
  - 支持长连接、socket复用以及并发请求
  - 自动管理连接状态，在合适的时候帮你断开连接或重新连接
  - 各种异常处理

- 屏蔽复杂的网络操作：
  - Socket生命周期管理，用长连接还是短连接？
  - 并发请求带来的复杂度增长
  - TCP和UDP在socket复用上的差异
- https://github.com/ysbcc/easy_sock



私有协议解决了

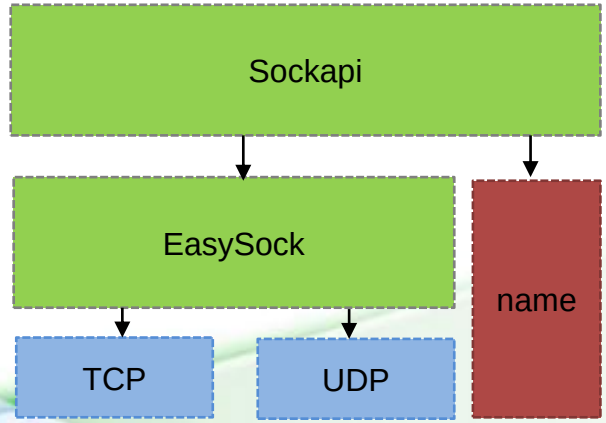
- 到这里，接入一个网络服务变的非常简单
  - 上层无需关心网络细节，只需实现包协议
  - 基于TCP长连接，支持并发请求，性能优越

search.js 444 Bytes

```
1  var search = require("../build/Release/obj.target/search_jce.node");
2  var apiFactory = require("txv.sockapi");
3
4
5  exports.create = function(cfg) {
6      return apiFactory({
7          keepAlive: cfg.keepAlive || false,
8          encode: search.encode,
9          decode: search.decode,
10         check: search.check,
11         l5modid: 322177 || cfg.l5modid //l5 modid, 找后台要。
12         l5cmd: 458752 || cfg.l5cmd //l5 cmd, 找后台要。
13     });
14 };
```

加入负载均衡

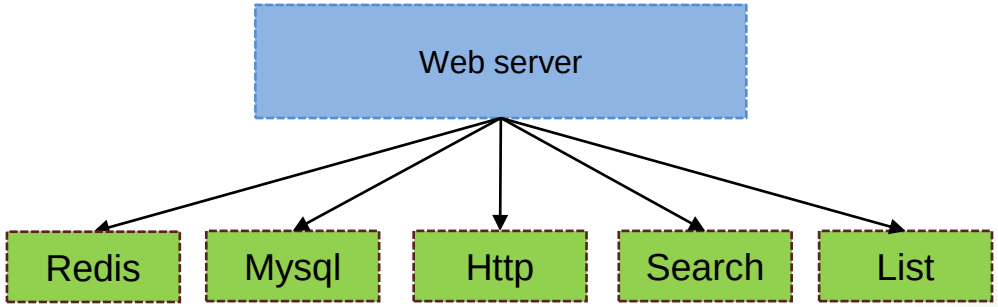
- 线上服务不能单点运行
- EasySock之上增加名字服务
  - 外部调用
  - 请求名字服务，获取服务器ip/端口
  - 创建EasySock建立服务器连接
  - 发起网络请求
  - 将连接保存进连接池，对不同ip地址的服务器各维护一个长连接



于是，可以愉快的写直出页面

- 每个服务需要调用很多个API接口

```
var mysql = require('mysql');
var redis = require('redis');
var request = require('request');
var protobuf = require('protobuf');
var jce = require('jce');
...
```



每个协议用起来还是不一样，香菇..

Step 2：进一步封装

- 开发：我不想知道协议细节，只想好好的渲染页面
- 怎样的写法最直观？

```
var redis = require("redis"),
    client = redis.createClient({
        host: '127.0.0.1',
        port: 6379,
        no_ready_check: true,
        return_buffers: true,
        retry_strategy: function (option) {
          if (option.attempt == 3) {
            return new Error('attempt over 3 times');
          }
          return 200;
    });

client.on("error", function (err) {
    console.log("Error " + err);
});

client.get('foo'), function(error, res){
    console.log(res);
    client.end();
});
```

```
var fetch = Fetcher.build({url: "redis://127.0.0.1:6379/foo"});
fetch().then(res => {...});
```

使用伪协议来声明请求

- 没有什么比url地址更简单明了
- 各种协议的请求变成url文本：

```
"mysql://user:password@host:port/database?
query=encodeURIComponent('select id from config where id=1')"
```

```
"redis://[{auth:pwd}@]{host}{port}/{key1}/{key2}/{key3}"
```

```
"searchservice://l5modid:l5cmd@testhost:testport?keyWord={keyWord}"
```

```
"http://v.qq.com"
```

套路二：数据请求器

- 把所有数据请求服务归一化成伪协议形式：

```
var client = redis.createClient();

//注册一个redis伪协议
fetcher.registerRequestor('redis', function(cfg) {
  var url = nodeurl.parse(cfg.url);

  return new Promise((resolve, reject)=> {
      client.get(url.pathname, function(err, res){
          err ? reject(err) : resolve(res);
      });
  })
});
```

前人种树
后人乘凉

```
//一个请求器实例
var fetch = Fetcher.build({
    url: "redis://127.0.0.1:6379/{key}"
});

//调用接口
fetch({key: "pagelet_0"}).then(res=> {
  console.log(res)
});
```

多路请求的声明

- 需要多个请求时，可以用json定义

```
{
    rank : {
        type: 'request',
        action: {
            url: "http://api.douban.com/v2/movie/us_box",
        }
    },
    detail : {
      type: 'request',
      action: {
          url: "http://api.douban.com/v2/movie/subject/{QUERY.id}"
      }
    },
    userinfo : {
      type: 'request',
      action: {
          url: "redis://myredis.com/{COOKIE.userid}"
      }
    },
}
```

- 所有返回数据都会放在一个object里，方便页面渲染

```
console.log(data.rank);
console.log(data.detail);
console.log(data.userinfo);
```

Step3：引入数据处理和流程控制

- 对于一个页面的数据请求，还有一些事情要处理：
  - 存在并行请求和串行请求，请求A可能依赖于请求B
  - 参数合法性校验
  - 数据处理
  - 错误处理

- dependencies：请求依赖的前置条件
- fixBefore：请求前处理
- fixAfter：请求后处理

```
{
    rank : {
        type: 'request',
        action: {
            url: "http://api.douban.com/v2/movie/us_box",
        }
    },
    detail : {
      type: 'request',
      dependencies: ["rank"]
      action: {
          url: "http://api.douban.com/v2/movie/subject/{id}"
          fixBefore: function(param){
            return param.rank.subjects[0];
          },
          fixAfter: function(data, param) {
            if (data.retcode != 0) {
              throw new Error('invalid result');
            }
          },
          onError: function(error) {

          }
      }
    }
}
```

套路三：一个完整的直出页面

- 所有页面生成都可归结为这3个过程：
  - 请求数据
  - 加工数据
  - 页面渲染

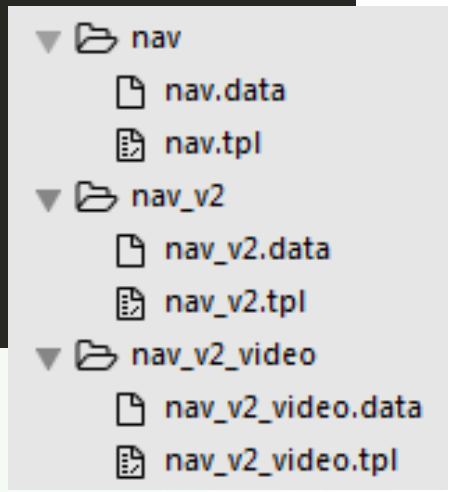
- 一个直出服务只需写2个文件
  - data.json 用于描述数据请求，以及数据处理
  - template.tpl 模板文件，用于渲染

```
var pigfarm = require('pigfarm.js');

var app = pigfarm({
    render: function(data) {
        return render(data, 'template.tpl');
    },
    data: require('data.json')
});
```

还没有结束

- 一旦框架建起来后，其他事情事半功倍
- 领导突然说要加缓存——写个cache请求器

```
{
  url: 'cache://somekey?expire=20&fresh=10&forceupdate={forceupdate}',
  update: {
    data: {
      rank: {
        type: 'request',
        action: "http://api.douban.com/v2/movie/us_box"
      }
    },
    render: function (data) {
      return `<h1>${data.rank}</h1>`;
    }
  },
  fixBefore: function(param) {
    return {forceupdate: !!param.query.update}
  }
});
```



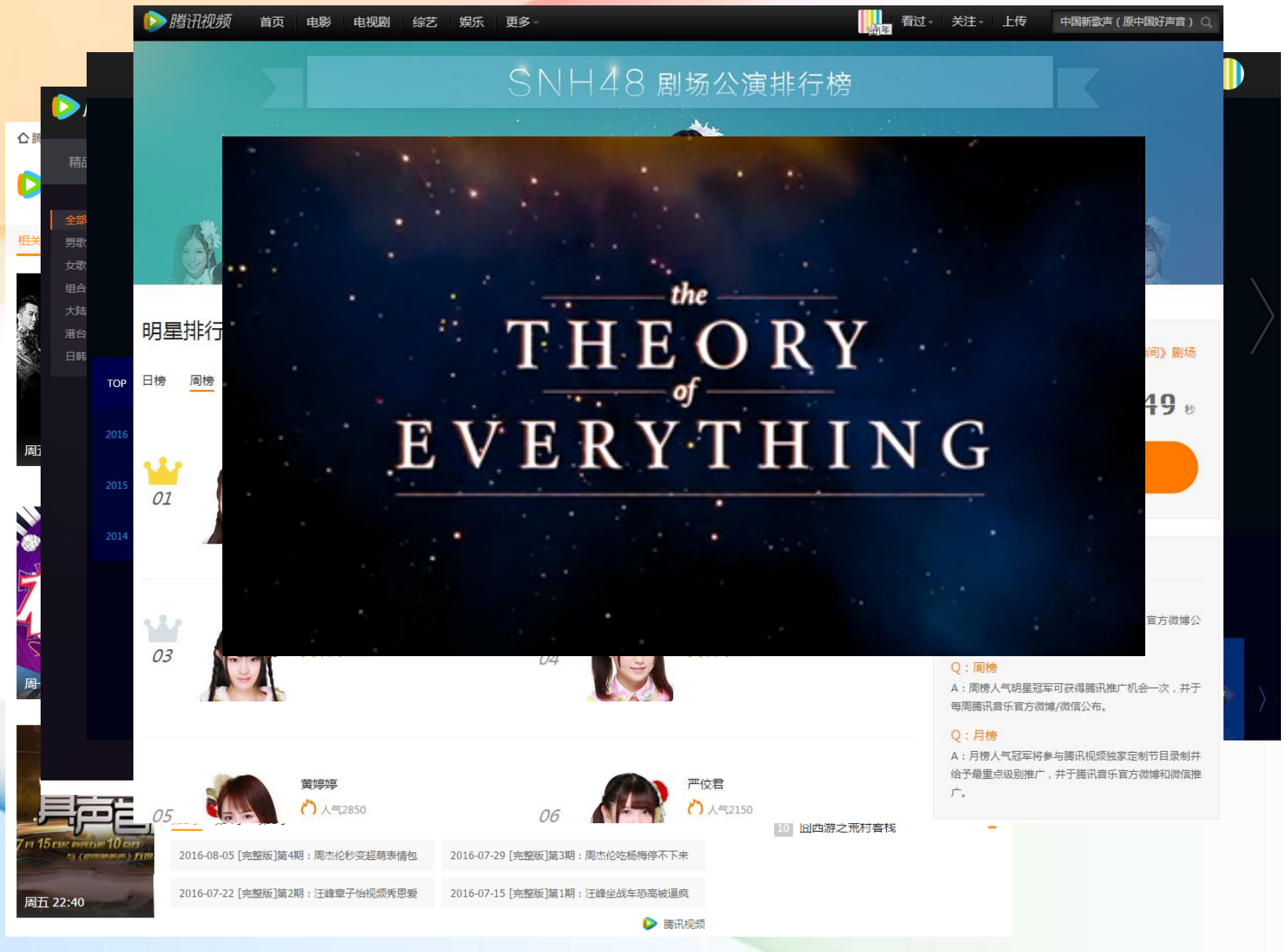
- 实现组件化，让“9个女人一个月生出孩子”成为可能

换个外壳，变成BigPipe

- 接收请求后第一时间先返回页面头部，再执行异步请求
- 以组件为单位，分段输出模板

```
var app = koa();
var chunker = require('auto-chunker')();

app.use(function *(){
    this.autoChunkerArray = [{
        data: {},
        template: '模块1.tpl'
    }, {
        data: {},
        template: '模块2.tpl'
    }, {
        data: {},
        template: '模块3.tpl'
    }]
});

app.use(chunker);
```

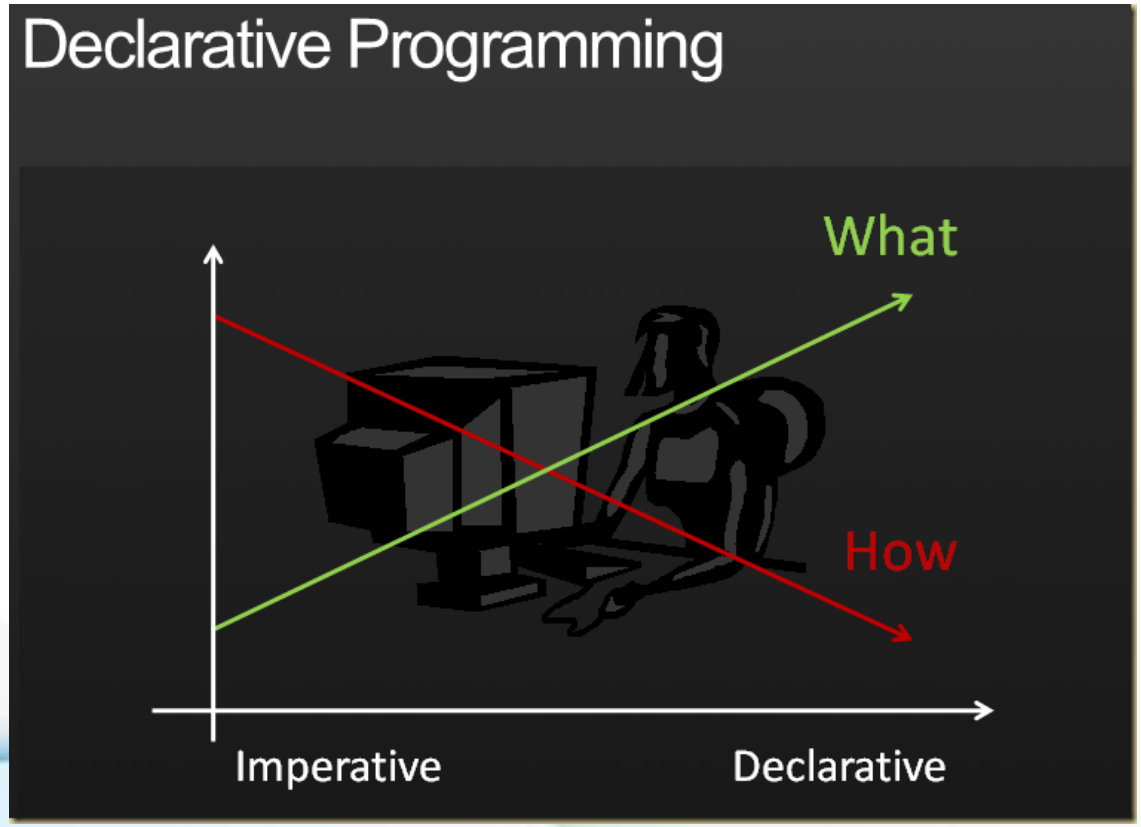
通过套路实现的页面


腾讯视频
首页
电影
电视剧
综艺
娱乐
更多
看过
关注
上传
中国新歌声（原中国好声音）
SNH48 剧场公演排行榜
the
THEORY
of
EVERYTHING
明星排行
日榜
周榜
TOP
2016
2015
2014
01
03
04
05
06
黄婷婷
人气2850
严佼君
人气2150
Q：周榜
A：周榜人气明星冠军可获得腾讯推广机会一次，并于每周腾讯音乐官方微博/微信公布。
Q：月榜
A：月榜人气冠军将参与腾讯视频独家定制节目录制并给予最重点级别推广，并于腾讯音乐官方微博和微信推广。
2016-08-05 [完整版]第4期：周杰伦秒变超萌表情包
2016-07-29 [完整版]第3期：周杰伦吃杨梅停不下来
2016-07-22 [完整版]第2期：汪峰章子怡视频秀恩爱
2016-07-15 [完整版]第1期：汪峰坐战车恐高被逼疯
腾讯视频
周五 22:40

这三个套路，实际上在做什么？

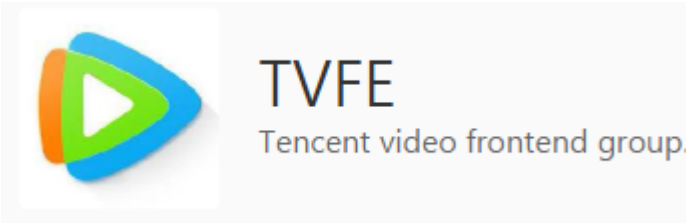
从命令式编程到声明式编程

- 业务开发不想关心过程（How），只想告诉程序要做什么（What）
- 相似的功能和逻辑只写一次
- 专注业务逻辑开发，其他不用操心
- 边际开发成本递减



欢迎关注腾讯视频前端开发团队

https://github.com/tvfe



谢 谢

腾讯 姚穗斌
ysbcc@qq.com