

# dubbo2.js

@hufeng

Takahashi Method

高橋流簡報法

很大

很透

很愉快

认真，老司机不等人

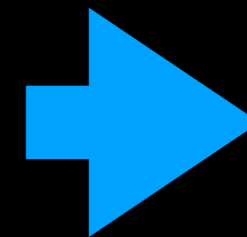
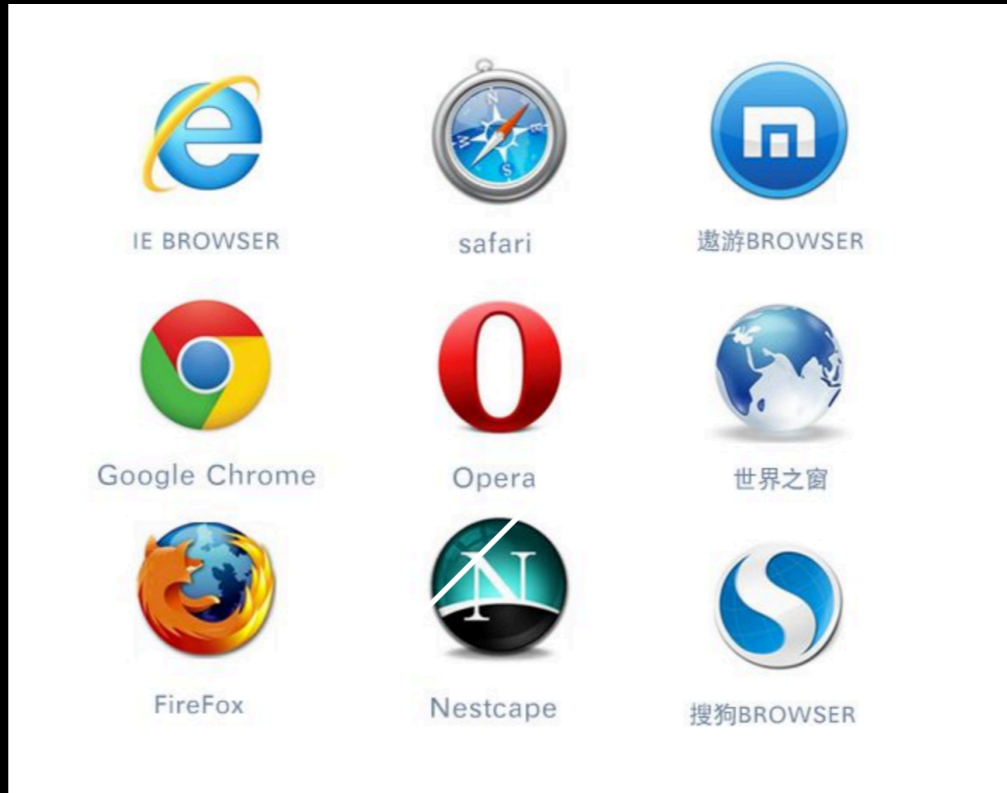
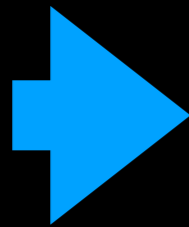


# Fronted-End development

# History



**xhtml**



**HTML**

CSS

**JS**

**Dw**

**Ps**

**Fl**

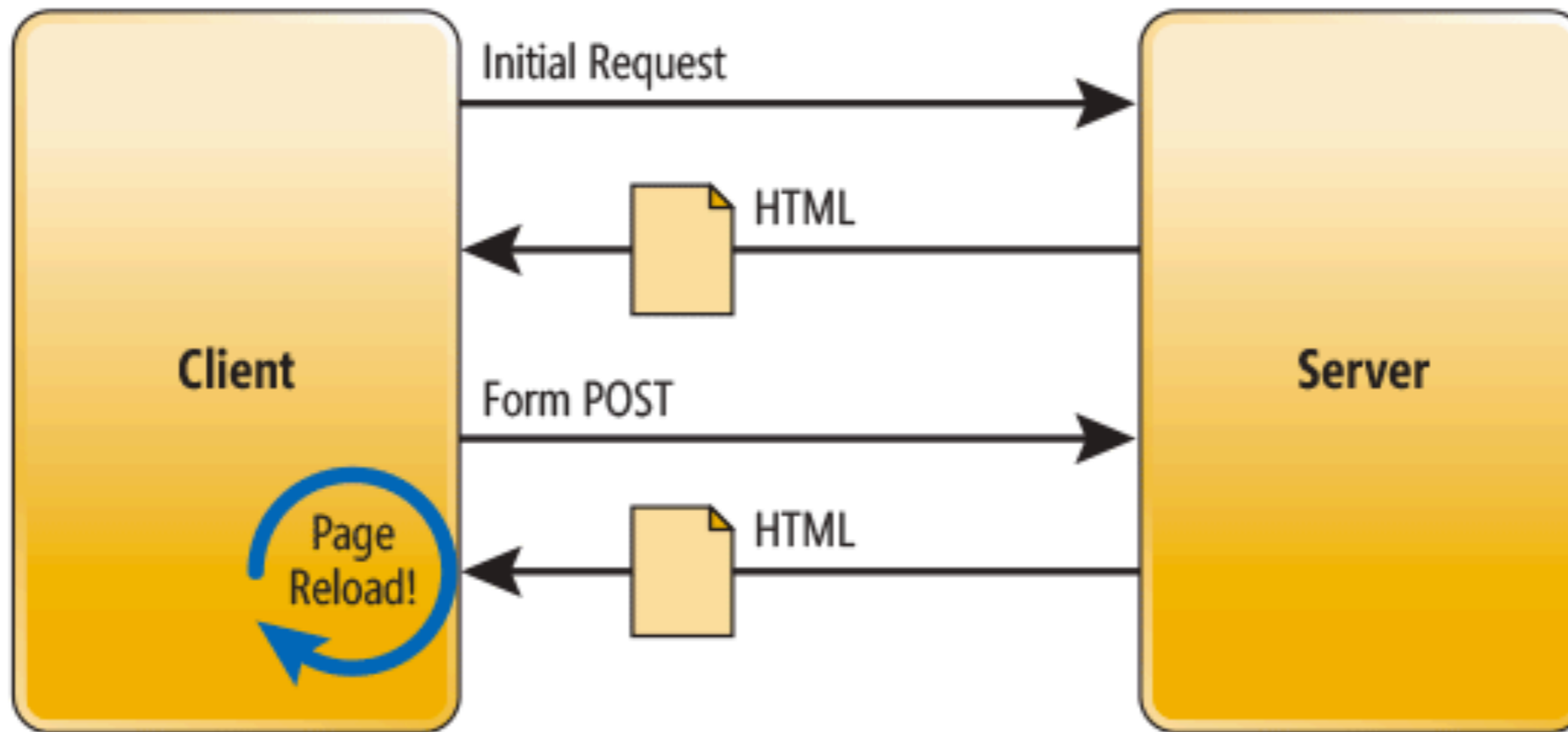
- 前端专注于内容的展示
- 解决浏览器的兼容性
- 开发模式简单
- 技术栈比较简单

前端痛点？

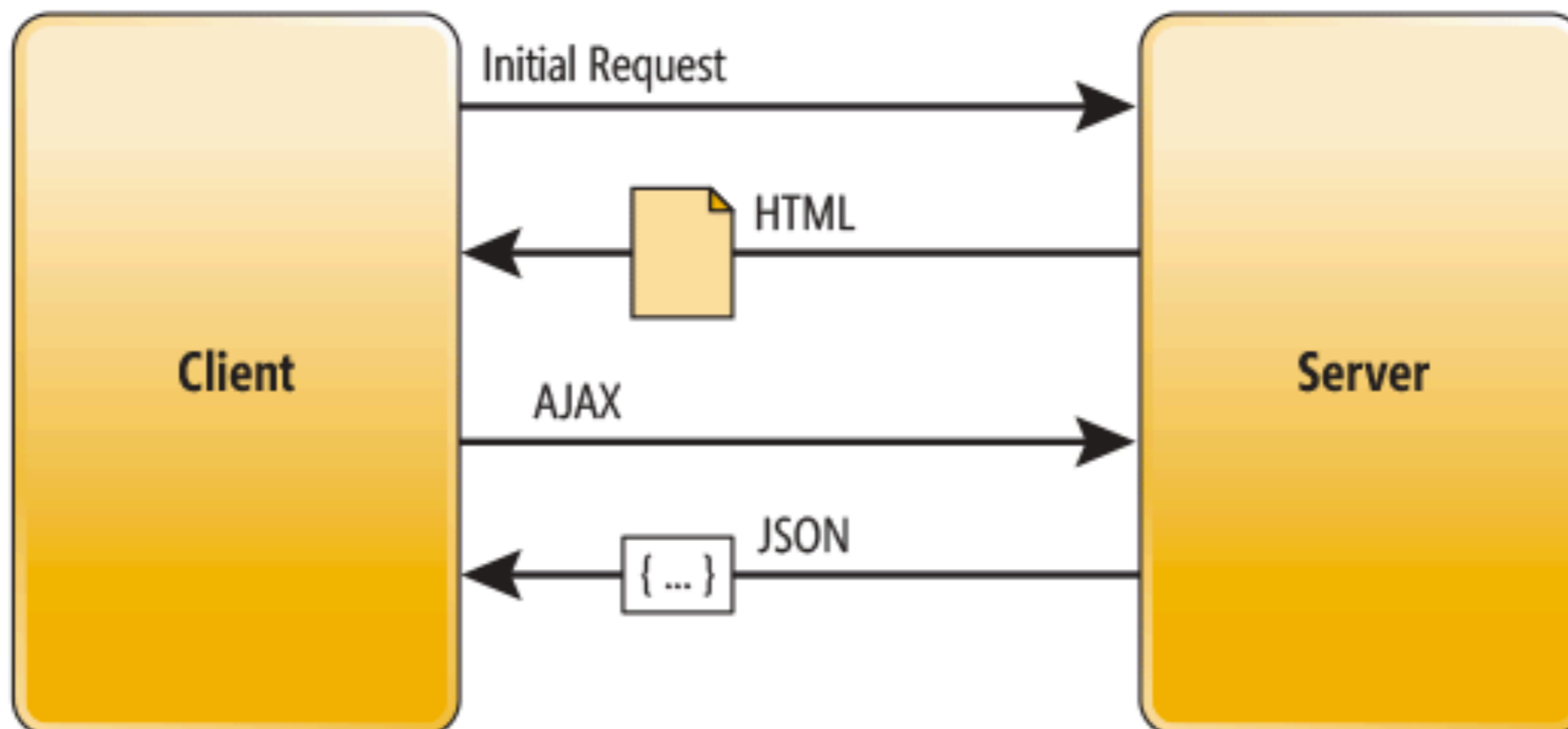
怎么成为后端!

**Ajax time**

## Traditional Page Lifecycle



## SPA Lifecycle





# The first leap

- 浏览器中的是应用不是内容
- 服务器交付的是数据而不是内容
- 用户体验变得流畅而连续

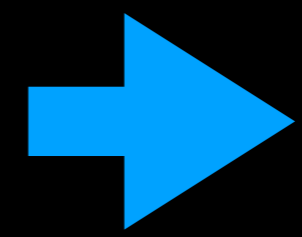
# Problem

- From web page |> web app (biz, interactive, data)
- a lot of javascript code(dependencies, module, reused)
- faster web

- 模块化(require.js, sea.js, commonjs, es6)
- 组件化(web component, react, vue)
- 工程化(fis, webpack, babel, ts, prettier)

**challenge**

广告机  
微信公众号 Mobile-web  
微信小程序 **多端** Web  
IoT设备  
Server iOS Android  
自动售货机



**产品**

Javascript  
HTML Typescript CSS  
Java **语言** Kotlin  
Swift Objective-c  
ReasonML

**设计**

BI  
**数据** GA  
黑客增长

**敏捷交付**

**调性**

累

累觉不爱 :(

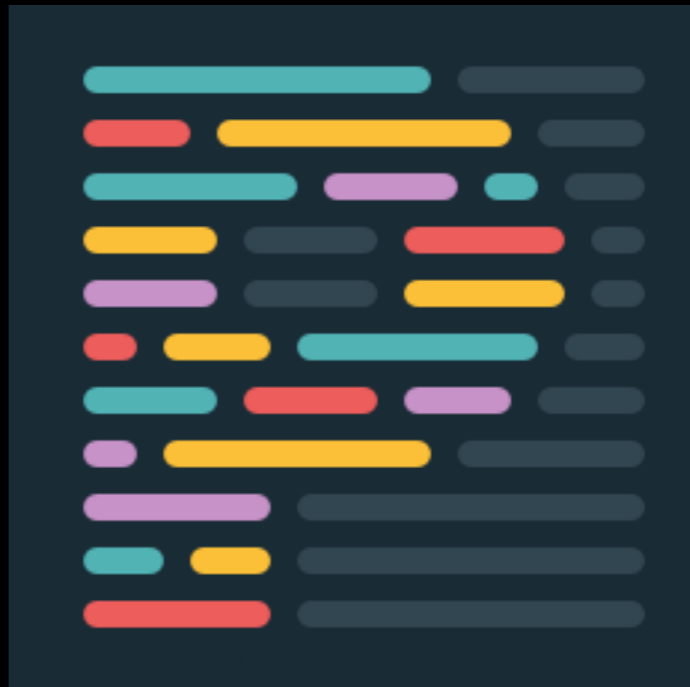
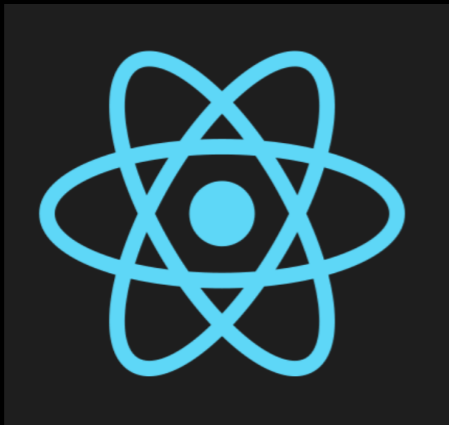
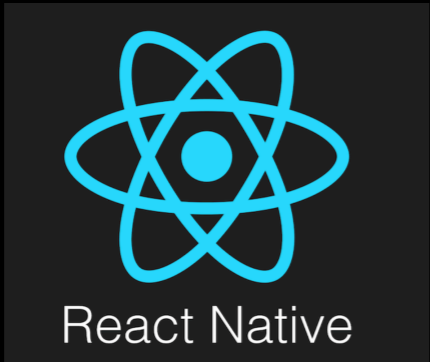
好消息，好消息

\*\*统统可以换不锈钢脸盆\*\*



“凡是能用JavaScript写出来的，  
最终都会用JavaScript写出来。”

–Atwood



beyond **php**

**javascript** is eating the world.

Any browser

Any Host

Any OS

多端时代的开始BFF开始萌芽

**BFF**了解一下!

- BFF(backend for Fronted)为前端专门设置服务端接口
- 从中心服务获取数据，组装，裁剪适配前端
- BFF转为前端服务，要能够响应前端的变化



# BFF

- BFF不是真正的BFF，仍是业务的controller
- 前端和BFF和中心沟通(部分前端参与java开发)
- Java相对Node开发体验更重
- 前端的SSR(Node + React)



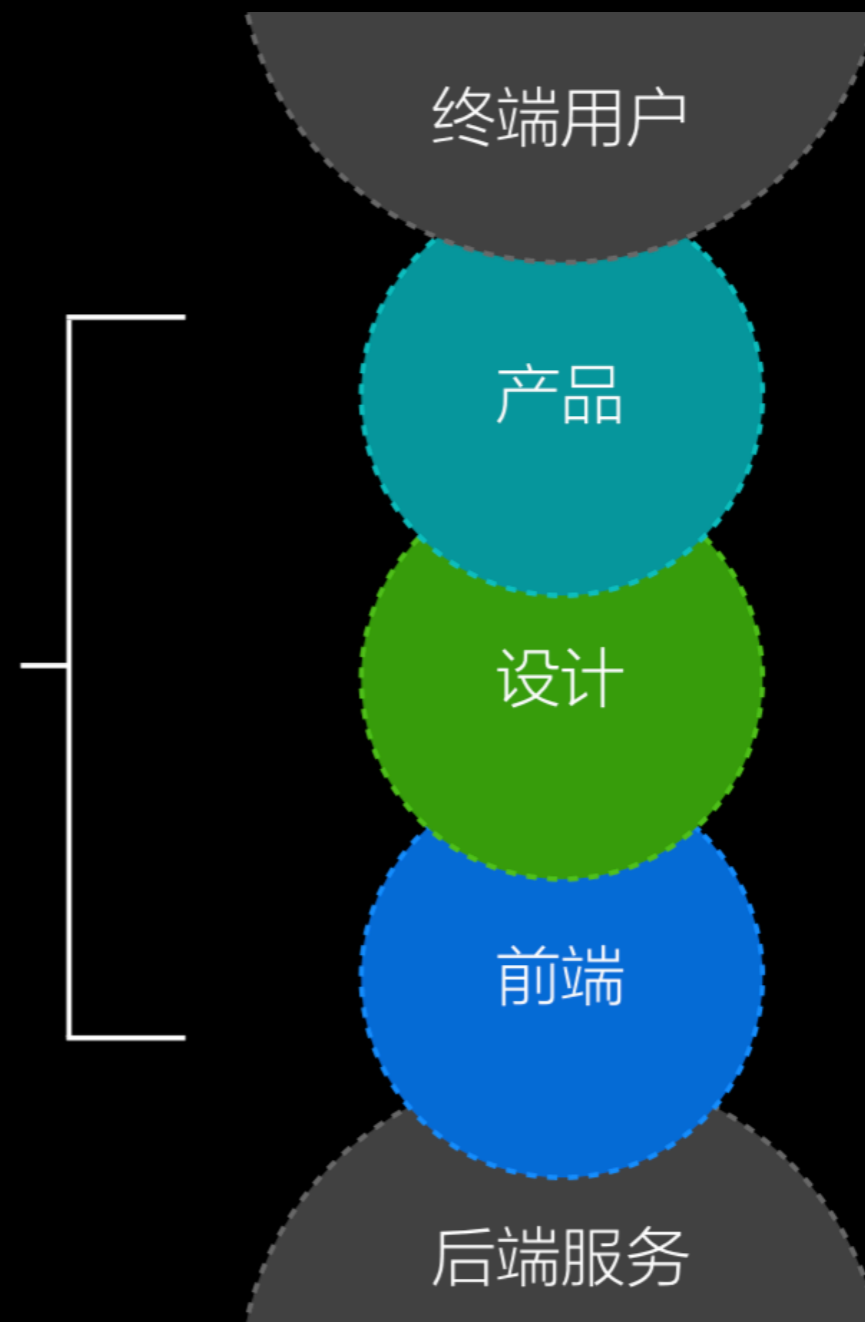
**Next Step: Node.JS BFF**

**从大前端到Service Team.**

# 前端技术是后端服务与人机界面的连接器



体验科技是技术与设计的融合，  
是服务与用户的连接。



终端用户



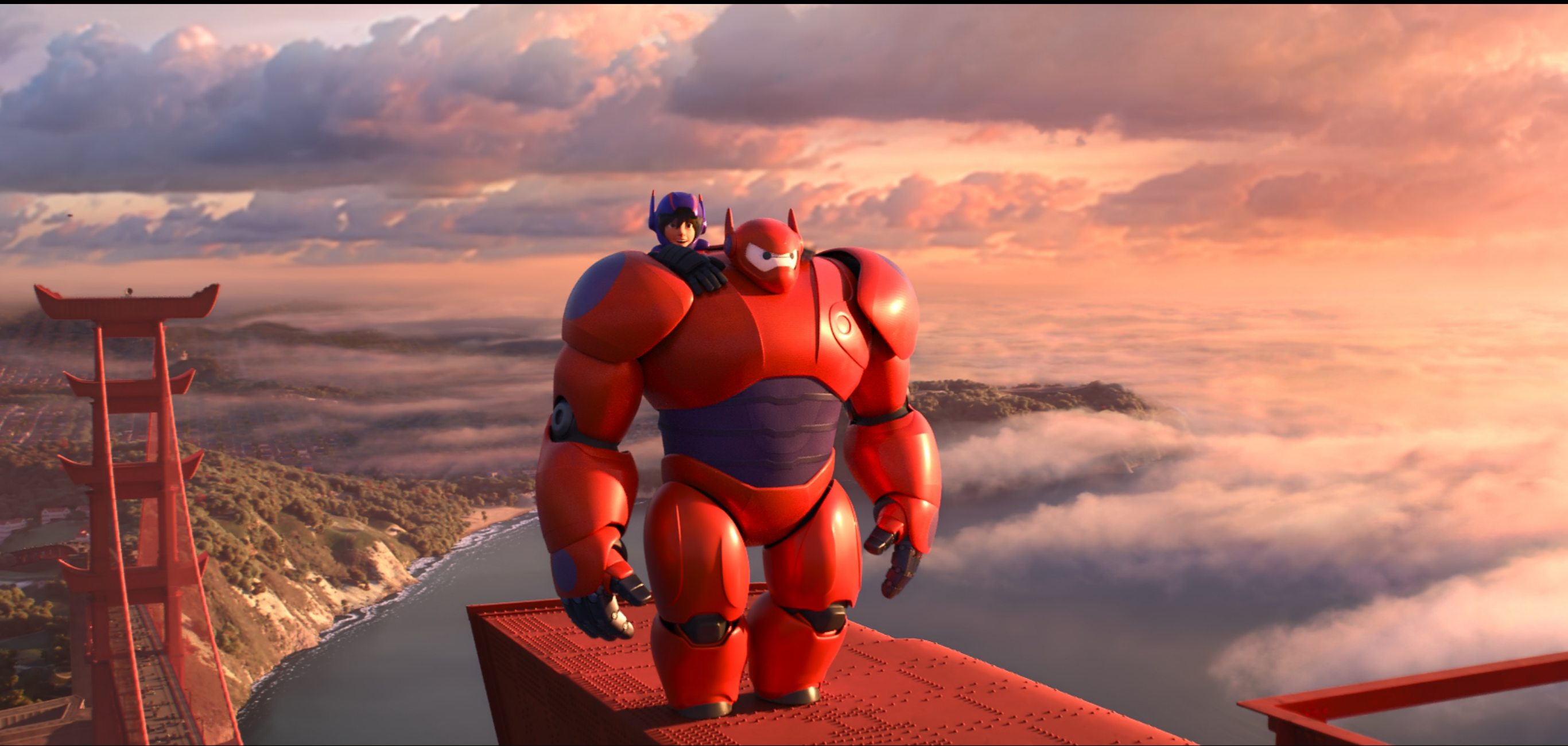
后端服务

$$UX = f(\text{services})$$

体验科技不止于人机界面的技术实现，  
更关乎终端用户的使用体验。

 **Agile Service Team** 

# 超能陆战队



现实



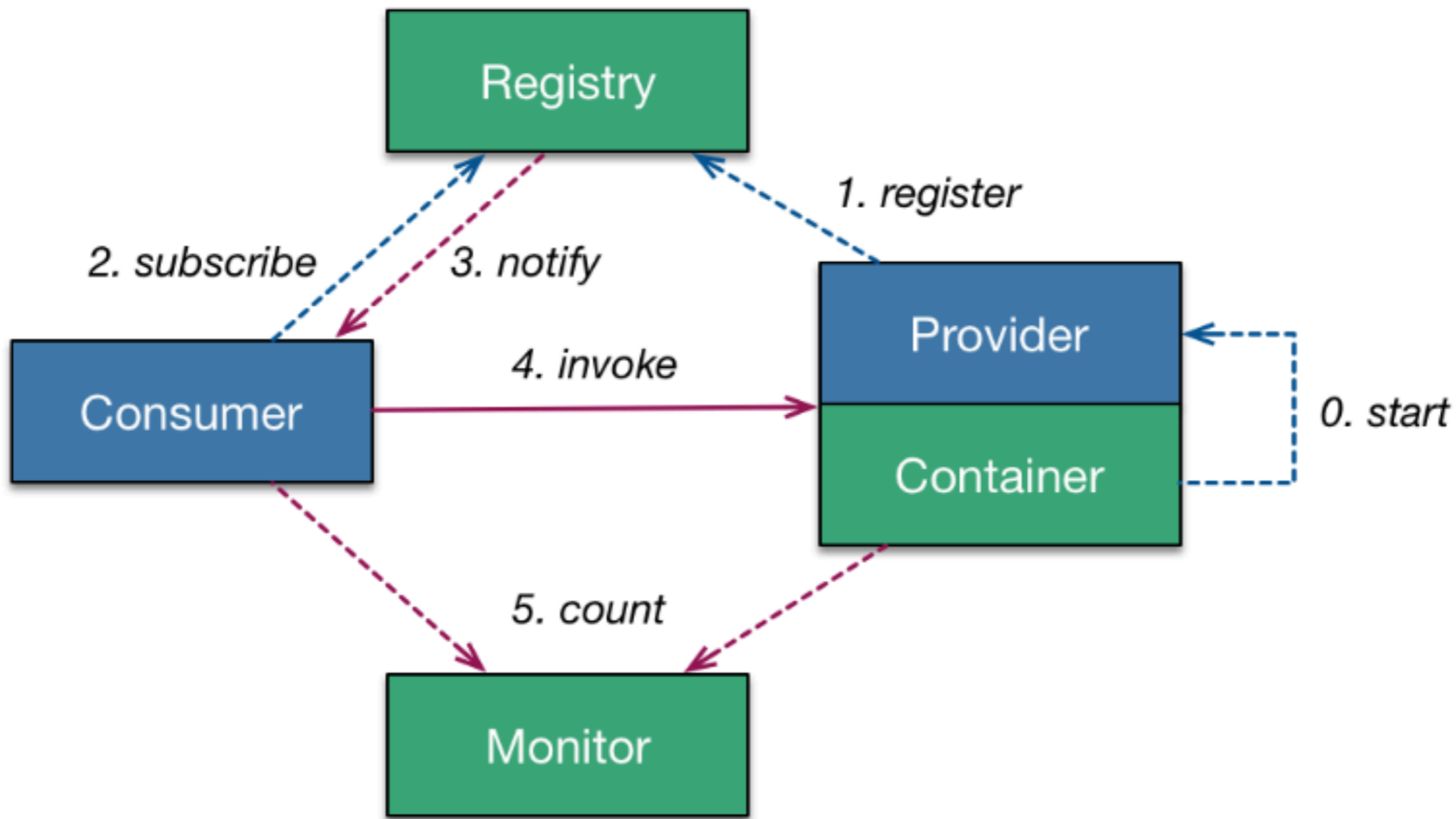
# Apache Dubbo (incubating)

Apache Dubbo™ (incubating) is a high-performance, java based, open source RPC framework.

[View on GitHub](#)

## Dubbo Architecture

---> init    - - - - -> async    ———> sync





Node Dubbo

JavaScript Java

dubbo-json-rpc

dubbo-client-py

node-dubbo-client

- 中心要暴露JSON-RPC
- 前端，后端没有推动起来，前端忙着解决多端挑战
- 调用不能很透明，开发体验不够友好

**node connect dubbo  
with native protocol**

Branch: master ▾


New pull request

Create new file

Upload files

Find file

Clone or download ▾

 hufeng add long

Latest commit 8bdbcb0 on May 31, 2017

📁 java	add long	11 months ago
📁 packages	add long	11 months ago
📄 .gitignore	add hessian web service	11 months ago
📄 LICENSE	Initial commit	11 months ago
📄 README.md	update readme	11 months ago
📄 lerna.json	init project	11 months ago
📄 package.json	init project	11 months ago

📖 README.md

# mega

我们天真的认为Node仍是BFF最好的解决方案，在开发效率和运行效率之间都是一个非常好的平衡。鉴于后端的Dubbo体系没有办法无缝的使用Node，这是天堑，但是可以克服的天堑。

**So this is mega project.**

1. node-hessian
2. node-dubbo
3. node-zookeeper

# Hessian

<http://hessian.caucho.com/doc/hessian-serialization.html>

```

top ::= value

binary
# 8-bit binary data split into 64k chunks
::= x41 b1 b0 <binary-data> binary # non-final chunk
::= 'B' b1 b0 <binary-data> # final chunk
::= [x20-x2f] <binary-data> # binary data of
# length 0-15
::= [x34-x37] <binary-data> # binary data of
# length 0-1023

boolean
# boolean true/false
::= 'T'
::= 'F'

class-def
# definition for an object (compact map)
::= 'C' string int string*

date
# time in UTC encoded as 64-bit long milliseconds since
# epoch
::= x4a b7 b6 b5 b4 b3 b2 b1 b0
::= x4b b3 b2 b1 b0 # minutes since epoch

double
# 64-bit IEEE double
::= 'D' b7 b6 b5 b4 b3 b2 b1 b0
::= x5b # 0.0
::= x5c # 1.0
::= x5d b0 # byte cast to double
# (-128.0 to 127.0)
::= x5e b1 b0 # short cast to double
::= x5f b3 b2 b1 b0 # 32-bit float cast to double

int
# 32-bit signed integer
::= 'I' b3 b2 b1 b0
::= [x80-xbf] # -x10 to x3f
::= [xc0-xcf] b0 # -x800 to x7ff
::= [xd0-xd7] b1 b0 # -x40000 to x3ffff

list
# list/vector
::= x55 type value* 'Z' # variable-length list
::= 'V' type int value* # fixed-length list
::= x57 value* 'Z' # variable-length untyped list
::= x58 int value* # fixed-length untyped list
::= [x70-77] type value* # fixed-length typed list
::= [x78-7f] value* # fixed-length untyped list

long
# 64-bit signed long integer
::= 'L' b7 b6 b5 b4 b3 b2 b1 b0
::= [xd8-xef] # -x08 to x0f
::= [xf0-xff] b0 # -x800 to x7ff
::= [x38-x3f] b1 b0 # -x40000 to x3ffff
::= x59 b3 b2 b1 b0 # 32-bit integer cast to long

```

```

map ::= 'M' type (value value)* 'Z' # key, value map pair
::= 'H' (value value)* 'Z' # untyped key, value

null
# null value
::= 'N'

object
# Object instance
::= 'O' int value*
::= [x60-x6f] value*

ref
# value reference (e.g. circular trees and graphs)
::= x51 int # reference to nth map/list/object

string
# UTF-8 encoded character string split into 64k chunk
::= x52 b1 b0 <utf8-data> string # non-final chunk
::= 'S' b1 b0 <utf8-data> # string of length
# 0-65535
::= [x00-x1f] <utf8-data> # string of length
# 0-31
::= [x30-x34] <utf8-data> # string of length
# 0-1023

type
# map/list types for OO languages
::= string # type name
::= int # type reference

value
# main production
::= null
::= binary
::= boolean
::= class-def value
::= date
::= double
::= int
::= list
::= long
::= map
::= object
::= ref
::= string

```

**Bool Int Date Null**

**Nested**



2015 D2

# js-to-java

---

Easy way to wrap js object to java object.

In [hessian.js](#), we need to write java classname with js object so make it encode as the write class.

## Install

---

```
$ npm install js-to-java
```

## Usage

---

### Example

```
var java = require('js-to-java');

// Java: com.java.Object o = new com.java.Object();
java('com.java.Object', { foo: 'bar' });
// => {$class: 'com.java.Object', $: { foo: 'bar' }}

// Java: Boolean r;
java.Boolean(true);
// => {$class: 'java.lang.Boolean', $: true}

// Java: short[] shorts = new short[] {1, 2, 3};
java.array('short', [1, 2, 3]);
// => {$class: '[short]', $: [1, 2, 3]}
```

&lt;&gt; Code

! Issues 2

🔗 Pull requests 10

📁 Projects 0

📖 Wiki

📊 Insights

js hessian binary web service protocol, support communicate with java

hessian

📌 215 commits

🔗 14 branches

📦 70 releases

👤 8 contributors

📄 MIT

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾



dead-horse Release 2.8.1

Latest commit 87ed8d6 on Jan 18

📁 benchmark	feat: support convert java.util.Locale to com.caucho.hessian.io.Local...	4 months ago
📁 lib	fix: compose cache key with class and fields length (#102)	3 months ago
📁 test	fix: compose cache key with class and fields length (#102)	3 months ago
📄 .autod.conf.js	fix: support writeLong parameter is a Long object (#96)	5 months ago
📄 .gitignore	fix: v2 list encode	2 years ago
📄 .jshintignore	refactor hessian writeObject with real java codes	4 years ago
📄 .jshintrc	feat: support cache class for v2/decode (#90)	6 months ago
📄 .travis.yml	Release 2.2.1	2 years ago
📄 AUTHORS	Release 2.1.9	2 years ago
📄 History.md	Release 2.8.1	3 months ago
📄 LICENSE	test: use npm scripts instead of Makefile	3 years ago
📄 README.md	feat: hessian2 optimize codec (#97)	4 months ago
📄 index.js	feat: support cache class for v2/decode (#90)	6 months ago
📄 package.json	Release 2.8.1	3 months ago

📖 README.md



6666

厉害了:)



科普

# RPC

- Remote Procedure Call 远程过程调用
- 透明，就像调用本地方法一样
- 不需要了解底层网络协议

# RPC

- Protocol (HTTP, HTTP2, TCP, UDP, QUIC, .....)
- Serializable(msgpack, hessian, protobuf, ....)
- IO(block IO, No-block IO(epoll), async io)

**Node**的IO是什么类型?



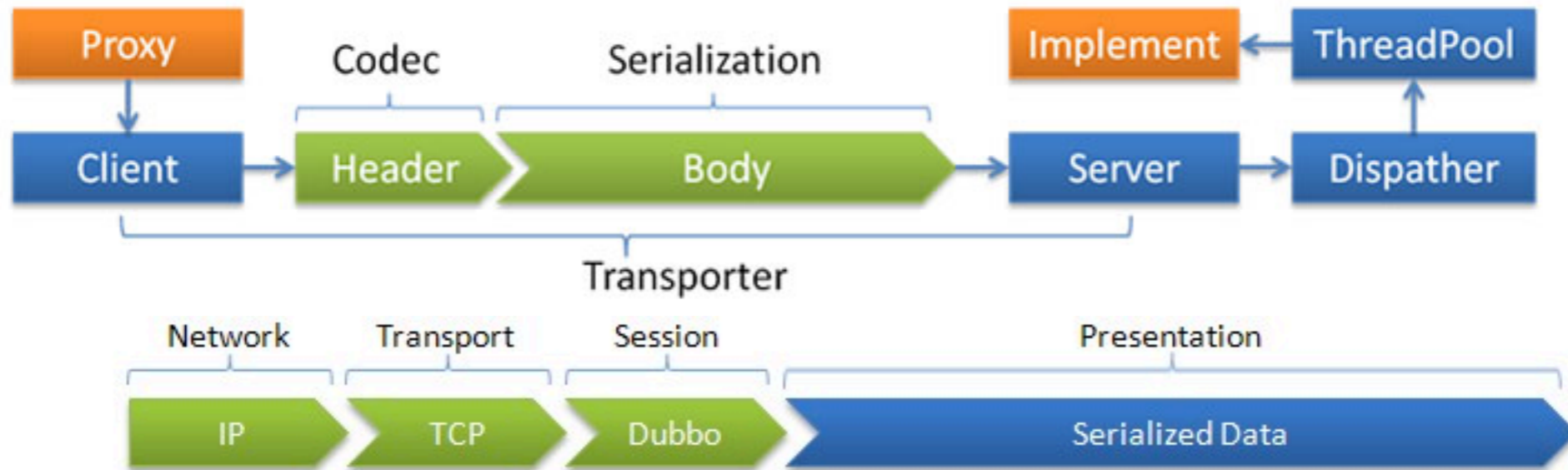
[HOME](#)[ABOUT](#)[DOWNLOADS](#)[DOCS](#)[GET INVOLVED](#)[SECURITY](#)[NEWS](#)[FOUNDATION](#)

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#). Node.js uses an event-driven, [non-blocking I/O model that makes it lightweight and efficient](#). Node.js' package ecosystem, [npm](#), is the largest ecosystem of open source libraries in the world.

# Dubbo RPC

- Protocol => TCP
- Serializable => Hessian
- IO => (No-Block IO)

# Dubbo Protocol



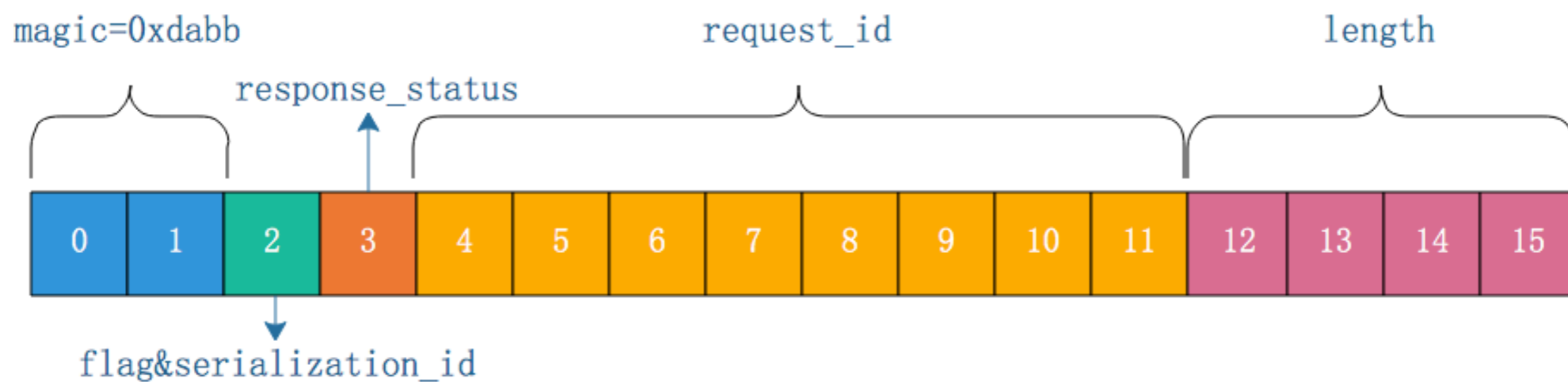
Bit offset	0 - 3	4 - 7	8 - 13	14 - 15	16 - 18	19 - 31
0	Version	Header Length	Differentiated Services Code Point	Explicit Congestion Notification	Total Length	
32	Identification				Flags	Fragment Offset
64	Time to Live	Protocol			Header Checksum	
96	Source IP Address					
128	Destination IP Address					
160	Options (if Header Length > 5)					

Bit offset	0 - 3	4 - 7	8	9	10	11	12	13	14	15	16 - 23	24 - 31
0	Source port					Destination port						
32	Sequence number											
64	Acknowledgment number (if ACK set)											
96	Data offset	Reserved	C W R	E C E	U R G	A C K	P R E	S S H	R S T	S Y N	F I N	Window Size
128	Checksum					Urgent pointer (if URG set)						
160	Options (if Data Offset > 5)											padding

Bit offset	0 - 7	8 - 15	16 - 20	21	22	23	24 - 31
0	Magic High	Magic Low	Serialization id	event	Two way	Req /res	status
32 - 95	Id (long)						
96 - 127	data length						

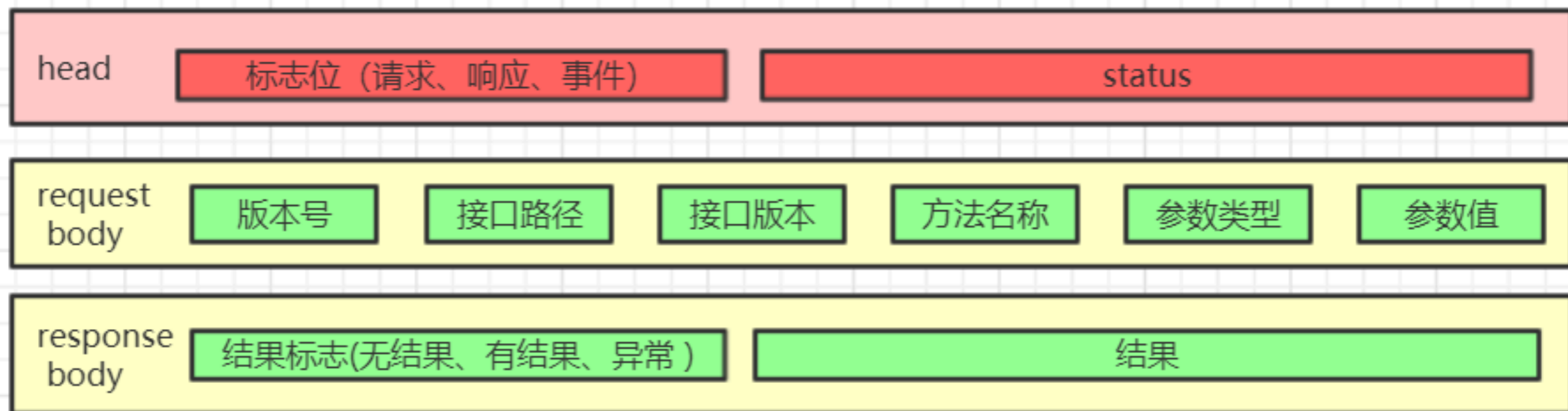
# Dubbo Header

Header结构



## Dubbo 报文格式

序列化



**Move fast break things.**

**Hack: baby step**



~

> ls

**Applications** **Desktop** **Downloads** **Hack** **Movies** **Pictures** **Workspace**  
**Cloud** **Documents** **Github** **Library** **Music** **Public** **Youtube**

~

> █



```
//dubbo的序列化协议
//com.alibaba.dubbo.remoting.exchange.codec.ExchangeCodec
//encodeRequest
```

```
public class ExchangeCodec extends TelnetCodec {

    // header length.
    protected static final int HEADER_LENGTH = 16;
    // magic header.
    protected static final short MAGIC = (short) 0xdabb;
    protected static final byte MAGIC_HIGH = Bytes.short2bytes(MAGIC)[0];
    protected static final byte MAGIC_LOW = Bytes.short2bytes(MAGIC)[1];
    // message flag.
    protected static final byte FLAG_REQUEST = (byte) 0x80;
    protected static final byte FLAG_TWOWAY = (byte) 0x40;
    protected static final byte FLAG_EVENT = (byte) 0x20;
    protected static final int SERIALIZATION_MASK = 0x1f;
    private static final Logger logger = LoggerFactory.getLogger(ExchangeCodec.class);

    protected void encodeRequest(Channel channel, ChannelBuffer buffer, Request req) throws
        Serialization serialization = getSerialization(channel);
        // header.
        byte[] header = new byte[HEADER_LENGTH];
        // set magic number.
        Bytes.short2bytes(MAGIC, header);

        // set request and serialization flag.
        header[2] = (byte) (FLAG_REQUEST | serialization.getContentTypeId());

        if (req.isTwoWay()) header[2] |= FLAG_TWOWAY;
        if (req.isEvent()) header[2] |= FLAG_EVENT;

        // set request id.
        Bytes.long2bytes(req.getId(), header, off: 4);

        // encode request data.
        int savedWriteIndex = buffer.writerIndex();
        buffer.writerIndex(savedWriteIndex + HEADER_LENGTH);
        ChannelBufferOutputStream bos = new ChannelBufferOutputStream(buffer);
        ObjectOutput out = serialization.serialize(channel.getUrl(), bos);
        if (req.isEvent()) {
            encodeEventData(channel, out, req.getData());
        } else {
            encodeRequestData(channel, out, req.getData());
        }
    }

    @Override
    protected void encodeRequestData(Channel channel, ObjectOutput out, Object data,
        RpcInvocation inv = (RpcInvocation) data;

        out.writeUTF(inv.getAttachment(Constants.DUBBO_VERSION_KEY, DUBBO_VERSION));
        out.writeUTF(inv.getAttachment(Constants.PATH_KEY));
        out.writeUTF(inv.getAttachment(Constants.VERSION_KEY));

        out.writeUTF(inv.getMethodName());
        out.writeUTF(ReflectUtils.getDesc(inv.getParameterTypes()));
        Object[] args = inv.getArguments();
        if (args != null)
            for (int i = 0; i < args.length; i++) {
                out.writeObject(encodeInvocationArgument(channel, inv, i));
            }
        out.writeObject(inv.getAttachments());
    }
}
```

```
    5-12个字节, 请求id
* 13-16个字节, 请求数据长度
*
* @param payload body的长度
*/
private encodeHead(payload: number) {
    //header
    const header = Buffer.alloc(DUBBO_HEADER_LENGTH);

    //set magic number
    //magic high
    header[0] = DUBBO_MAGIC_HEADER >>> 8;
    //magic low
    header[1] = DUBBO_MAGIC_HEADER & 0xff;

    // set request and serialization flag.
    header[2] = FLAG_REQUEST | HESSIAN2_SERIALIZATION_CONTENT_ID | FLAG_TWOWAY;

    //requestId
    this.setRequestId(header);

    //check body length
    if (payload > 0 && payload > DUBBO_DEFAULT_PAY_LOAD) {
        throw new DubboEncodeError(
            `Data length too large: ${payload}, max payload: ${DUBBO_DEFAULT_PAY_LOAD}`,
        );
    }

    //body长度int-> 4个byte
    const bodyLengthBuff = binaryNum(payload, 4);
    header[12] = bodyLengthBuff[0];
    header[13] = bodyLengthBuff[1];
    header[14] = bodyLengthBuff[2];
}
```

```
private encodeBody() {
  //hessian v2
  const encoder = new Hessian.EncoderV2();

  const { ...
} = this._ctx;

  //dubbo version
  encoder.write(dubboVersion);
  //path interface
  encoder.write(dubboInterface);
  //interface version
  encoder.write(version);
  //method name
  encoder.write(methodName);
  //parameter types
  encoder.write(DubboEncoder.getParameterTypes(methodArgs));

  //arguments
  if (methodArgs && methodArgs.length) {
    for (let arg of methodArgs) {
      encoder.write(arg);
    }
  }
}
```

```
/**
 * Dubbo codec.
 *
 * @author qianlei
 * @author chao.liuc
 */
public class DubboCodec extends ExchangeCodec implements Codec2 {

    public static final String NAME = "dubbo";
    public static final String DUBBO_VERSION = Version.getVersion(DubboCodec.class, Version.get
    public static final byte RESPONSE_WITH_EXCEPTION = 0;
    public static final byte RESPONSE_VALUE = 1;
    public static final byte RESPONSE_NULL_VALUE = 2;
    public static final Object[] EMPTY_OBJECT_ARRAY = new Object[0];
    public static final Class<?>[] EMPTY_CLASS_ARRAY = new Class<?>[0];
    private static final Logger log = LoggerFactory.getLogger(DubboCodec.class);

    @Override
    protected void encodeResponseData(Channel channel, ObjectOutputStream out, Object data) throws IOException {
        Result result = (Result) data;

        Throwable th = result.getException();
        if (th == null) {
            Object ret = result.getValue();
            if (ret == null) {
                out.writeByte(RESPONSE_NULL_VALUE);
            } else {
                out.writeByte(RESPONSE_VALUE);
                out.writeObject(ret);
            }
        } else {
            out.writeByte(RESPONSE_WITH_EXCEPTION);
            out.writeObject(th);
        }
    }
}
```

```
48 //com.alibaba.dubbo.remoting.exchange.codec.ExchangeCodec.encodeResponse/decode
49 export function decode<T>(bytes: Buffer): IDubboResponse<T> {
50     let res = null;
51     let err = null;
52
53     // set request and serialization flag.
54     const requestIdBuff = Buffer.alloc(8);
55     requestIdBuff[0] = bytes[4];
56     requestIdBuff[1] = bytes[5];
57     requestIdBuff[2] = bytes[6];
58     requestIdBuff[3] = bytes[7];
59     requestIdBuff[4] = bytes[8];
60     requestIdBuff[5] = bytes[9];
61     requestIdBuff[6] = bytes[10];
62     requestIdBuff[7] = bytes[11];
63
64     const requestId = convertBinaryNum(requestIdBuff, 8);
65     log(`decode parse requestId: ${requestId}`);
66
67     // const typeId = bytes[2];
68
69     // get response status.
70     const status = bytes[3];
71
72     log( ...
76     );
77
78     if (status !== DUBBO_RESPONSE_STATUS.OK) { ...
84     }
85
86     //com.alibaba.dubbo.rpc.protocol.dubbo.DecodeableRpcResult
87     const body = new HessianDecoderV2(bytes.slice(HEADER_LENGTH));
```

# 二进制bit的操作

获取最后一个字节 `0x0a4f & 0xff`

`<<`    `<<<`    `>>>`    `>>`

循环 `>> 32`

Javascript magic

```
parseInt((10).toString(2), 2);
```

**One week : toy**

**No long connection**

**One Node Socket**

**No heartbeat      No registry**

**Not support concurrent**

**Only Serializable**

工程化



# Mono-Repo



# Lerna

A tool for managing JavaScript projects with multiple packages.

npm v2.11.0 travis passing build passing  Slack Status

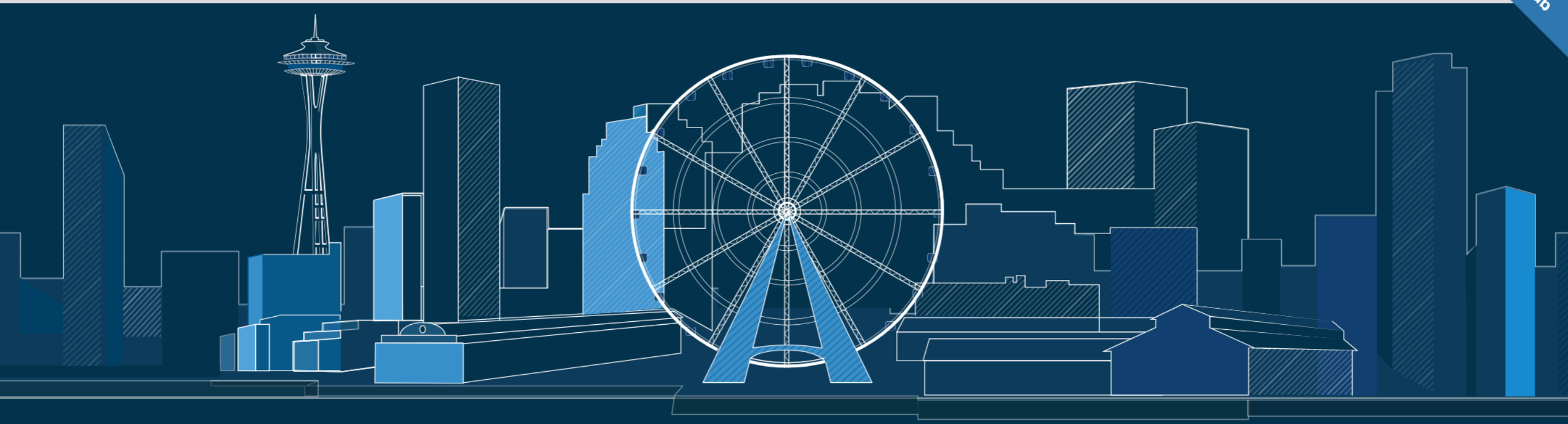
```
~/Github/dubbo2.js master
```

```
> tree -L 2 -I "node_modules|resources|java"
```

```
.
├── CONTRIBUTING.md
├── LICENSE
├── Makefile
├── README.md
├── _config.yml
├── dubbo.json
├── examples
│   ├── hello-egg
│   └── hello-koa
├── lerna.json
├── package.json
├── packages
│   ├── dubbo
│   ├── dubbo-invoker
│   ├── interpret-cli
│   └── interpret-util
├── tsconfig.json
└── yarn.lock
```

```
8 directories, 10 files
```

TypeScript 2.8 is now available. [Download](#) our latest version today!



# TypeScript

**JavaScript that scales.**

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.

Any browser. Any host. Any OS. Open source.

[Download](#)[Documentation](#)

## TypeScript configuration

```
{  
  "compilerOptions": {  
    "module": "commonjs",  
    "target": "es2017",  
    "moduleResolution": "node",  
    "esModuleInterop": true  
  }  
}
```

```
private _watch(providerPath: string, dubboInterface: string) {  
  // @ts-ignore  
  return async (e: zookeeper.Event) => {  
    log(`trigger watch ${providerPath}, type: %s`, e.getName());  
    const providers =  
      (await this._getProviderList(providerPath, dubboInterface)) || [];  
    const providerList = providers.map(ZkClient.parseUrl);  
    log(  
      'update dubboInterface % providerList %0',  
      dubboInterface,  
      providerList,  
    );  
  };  
}
```

```
1  {  
2    "extends": "../../tsconfig.json",  
3    "compilerOptions": {  
4      "rootDir": "./src",  
5      "outDir": "./es7",  
6      "declaration": true,  
7      "declarationDir": "./es7/typings",  
8      "noUnusedLocals": true,  
9      "noUnusedParameters": true  
10   },  
11   "exclude": ["./src/__tests__/**"]  
12 }  
13
```

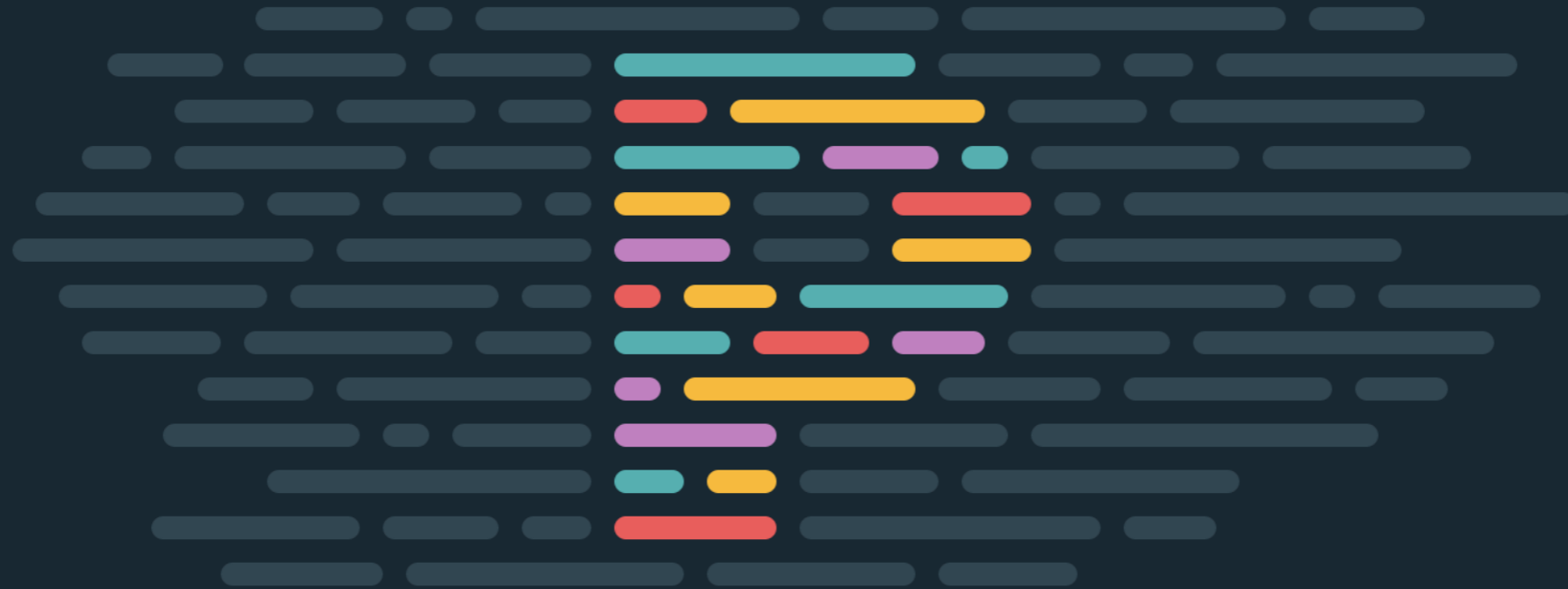
# 代码即文档

# 类型即文档

```
type TRequestId = number;
```

```
//请求队列
```

```
private readonly _requestQueue: Map<TRequestId, Context>
```



[TRY IT OUT](#)

[GET STARTED](#)

[OPTIONS](#)

## # What is Prettier?

- \* An opinionated code formatter
- \* Supports many languages
- \* Integrates with most editors
- \* Has few options

## # Why?

- \* You press save and code is formatted
- \* No need to discuss style in code review
- \* Saves you time and energy
- \* And more



# Jest



Delightful JavaScript Testing

[TRY OUT JEST](#)

[GET STARTED](#)

[WATCH TALKS](#)

[LEARN MORE](#)

★ Star 17,184



## Developer Ready

Complete and ready to set-up JavaScript testing solution. Works out of the box for any React project.



## Instant Feedback

Fast interactive watch mode runs only test files related to changed files and is optimized to give signal quickly.



## Snapshot Testing

Capture snapshots of React trees or other serializable values to simplify testing and to analyze how state changes over time.



```
// Jest Snapshot v1, https://goo.gl/fbAQLP
```

```
exports[`dubbo logger test dubbo logger middleware 1`] = `
Object {
  "costTime": 50,
  "exceptionMsg": "",
  "invokeTime": "2018-03-26 12:04:55",
  "logger": "dubboLogger",
  "methodName": "getUserInfo",
  "paramTypes": "[\\"com.alibaba.dubbo.demo.UserRequest\\"]",
  "paramValues": "[{\\"id\\":1,\\"name\\":\\"nodejs\\",\\"email\\":\\"node@qianmi.com\\"}]",
  "pinpointTraceId": "",
  "receiverHost": "172.19.36.1:20881",
  "receiverName": "com.alibaba.dubbo.demo.DemoService",
  "resultValue": "{\\"status\\":\\"ok\\",\\"info\\":{\\"name\\":\\"test\\",\\"id\\":\\"1\\"}}",
  "senderHost": "172.19.36.1",
  "senderName": "@qianmi/node-dubbo",
  "sessionId": "",
  "srvGroup": "",
  "uuId": "d6aea000-4906-11e8-914f-c3129335a91f",
  "version": "1.0.0",
}
`;
```

# Debug

~/Workspaces/dubbo2.js/examples/hello-koa

~/Workspaces/dubbo2.js/examples/hello-koa master\*

> ls

```
dubbo-es6.js      es6-http.js      node_modules      package.json      yarn.lock
dubbo.js          es7-http.js      package-lock.json server.js
```

~/Workspaces/dubbo2.js/examples/hello-koa master\*

> █

I

~  
> █

**DDD**

**Zookeeper**

**SocketPool**

**SocketWorker**

**Scheduler**

**Async model?**

**Queue**

//Queue真心好用

**Big Bug**  **嫩**

`ab -n 10000 -c 100`





**Buff-decode**



# ELK Logger

**Kafka-client**

**Egg-qm-logger**

**egg-log-tracer**

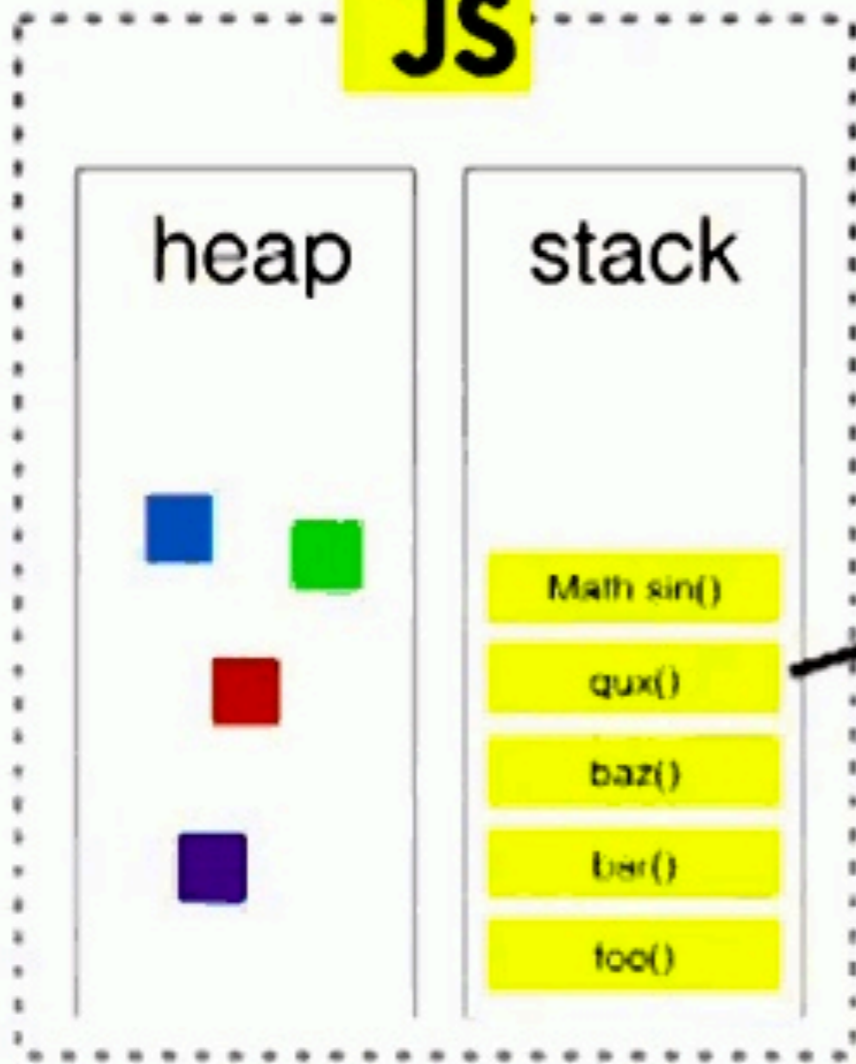
# Tracing

Request -> ? -> ?

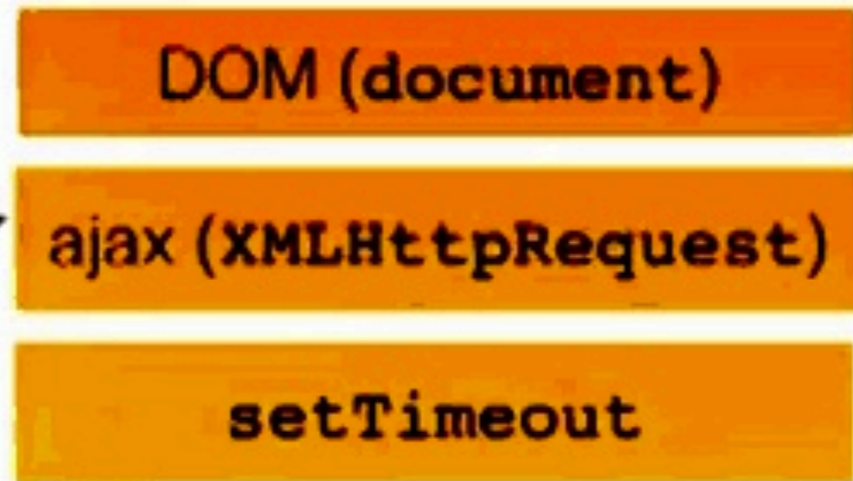
# Java ThreadLocal

# JavaScript EventLoop





## WebAPIs



event loop 

callback queue



# Zone.js

---

build passing cdnjs v0.8.26

Implements *Zones* for JavaScript, inspired by [Dart](#).

If you're using zone.js via unpkg (i.e. using `https://unpkg.com/zone.js`) and you're using any of the following libraries, make sure you import them first

- 'newrelic' as it patches global.Promise before zone.js does
- 'async-listener' as it patches global.setTimeout, global.setInterval before zone.js does
- 'continuation-local-storage' as it uses async-listener

## NEW Zone.js POST-v0.6.0

---

See the new API [here](#).

Read up on [Zone Primer](#).

## What's a Zone?

---

A Zone is an execution context that persists across async tasks. You can think of it as [thread-local storage](#) for JavaScript VMs.

See this video from ng-conf 2014 for a detailed explanation:



**But But But *async/await*.....**

**async\_hooks**

```
const async_hooks = require('async_hooks');

// Return the ID of the current execution context.
const eid = async_hooks.executionAsyncId();

// Return the ID of the handle responsible for triggering the callback of the
// current execution scope to call.
const tid = async_hooks.triggerAsyncId();

// Create a new AsyncHook instance. All of these callbacks are optional.
const asyncHook =
    async_hooks.createHook({ init, before, after, destroy, promiseResolve });

// Allow callbacks of this AsyncHook instance to call. This is not an implicit
// action after running the constructor, and must be explicitly run to begin
// executing callbacks.
asyncHook.enable();

// Disable listening for new asynchronous events.
asyncHook.disable();

//
// The following are the callbacks that can be passed to createHook().
//
```

```
1 import async_hooks from 'async_hooks';
2 import debug from 'debug';
3 const log = debug('dubbo:zone');
4
5 //alias type
6 export type AsyncId = number;
7 export type RootAsyncId = number;
8
9 /**
10  * ZoneContext 期待Zone的规范早日落地
11  */
12 export class ZoneContext {
13   constructor() {
14     log('init ZoneContext');
15     this.rootMap = new Map();
16     this.statckFrameMap = new Map();
17     this.initAsyncHook();
18   }
19
20   private rootMap: Map<AsyncId, object>;
21   private statckFrameMap: Map<AsyncId, RootAsyncId>;
```

```
npm install @qianmi/zone-context
```

**logstash-\***

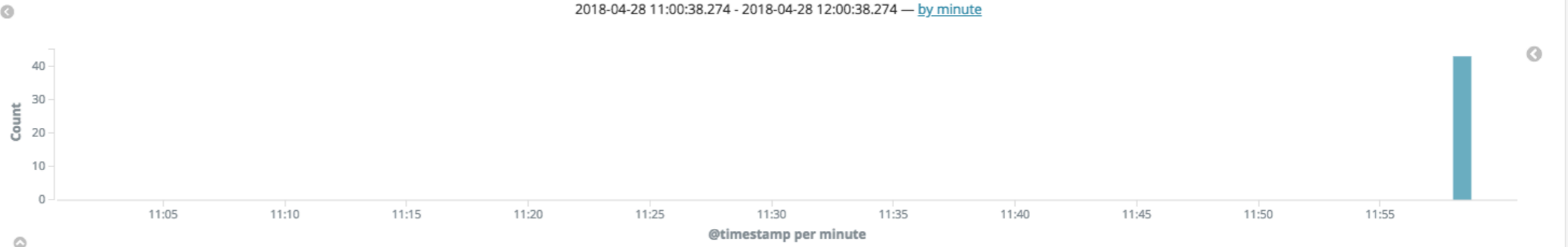
Selected Fields

- # costTime
- † receiverHost
- † uri
- # cost
- † receiverName

Available Fields

Popular

- † appName
- † exceptionMsg
- † host
- † level
- † methodName
- † paramValues
- † senderHost
- † senderName
- ⊙ @timestamp
- † @version
- † \_id
- † \_index
- # \_score
- † \_type
- † fromIp
- † invokeTime



Time	cost	costTime	uri	receiverHost	receiverName
2018-04-28 11:58:52.884	448	-	/cart/v3/confirmInitValidate?time=1524887932285	-	-
2018-04-28 11:58:52.884	-	14	-	172.17.16.110:20880	com.qianmi.cloudshop.api.marketing.d2p.D2pMarketingQueryProvider
2018-04-28 11:58:52.869	-	41	-	172.17.17.99:20880	com.qianmi.pc.item.api.brand.BrandQueryProvider
2018-04-28 11:58:52.843	-	3	-	172.17.11.109:20880	com.qianmi.pc.item.api.brand.BrandQueryProvider
2018-04-28 11:58:52.840	-	11	-	172.17.16.136:20880	com.qianmi.pc.price.api.PriceQueryProvider
2018-04-28 11:58:52.829	-	47	-	172.17.17.99:20880	com.qianmi.pc.item.api.sku.SkuQueryProvider
2018-04-28 11:58:52.828	-	7	-	172.17.16.136:20880	com.qianmi.pc.price.api.PriceQueryProvider
2018-04-28 11:58:52.826	-	2	-	172.17.17.99:20880	com.qianmi.pc.item.api.brand.BrandQueryProvider
2018-04-28 11:58:52.824	-	11	-	172.17.10.204:20880	com.qianmi.pc.price.api.PriceQueryProvider
2018-04-28 11:58:52.823	-	3	-	172.17.11.109:20880	com.qianmi.pc.item.api.brand.BrandQueryProvider
2018-04-28 11:58:52.821	-	7	-	172.17.17.99:20880	com.qianmi.pc.item.api.brand.BrandQueryProvider
2018-04-28 11:58:52.821	-	33	-	172.17.17.99:20880	com.qianmi.pc.item.api.sku.SkuQueryProvider



**Extention**

# MicroKernel

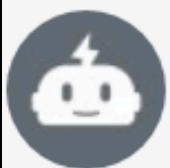
**Middleware**

# Koa-Compose + Context

```
//cost-time middleware
dubbo.use(async (ctx, next) => {
  const startTime = Date.now();
  await next();
  const endTime = Date.now();
  console.log(endTime - startTime);
});
```

```
dubbo.use(  
  dubboInvoke(  
    matcher  
      //精确匹配接口  
      .match('com.alibaba.demo.UserProvider', {  
        version: '1.0.0',  
        group: 'user',  
      })  
      //正则匹配  
      .match(/$com.alibaba.dubbo/, {  
        version: '2.0.0',  
        group: '',  
      })  
      //match thunk  
      match((ctx) => {  
        //computed....  
        return true  
      })  
  )  
)
```

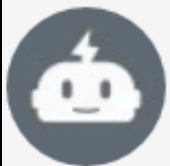
**Plugin**



dubbo2.js

机器人

dubbo was connected successfully. with registry 172.19.67.126:2181



dubbo2.js

机器人

d2p-visitor-bff:dubbo was connected successfully. with registry  
172.19.67.126:2181





**3ks Dubbo** 

**Apache Licence 2.0**

**Comments(en)**

**Remove Author**

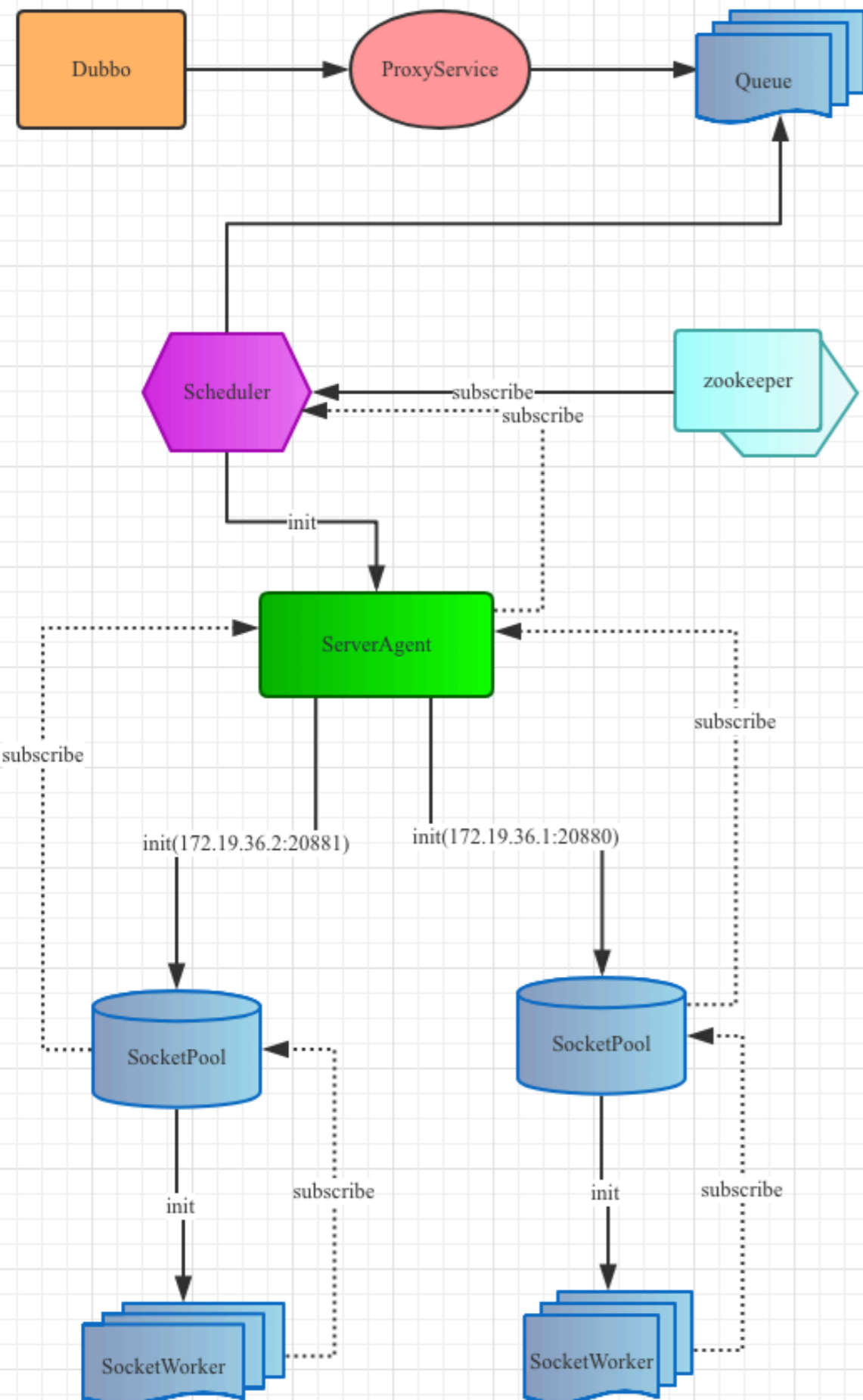
**smooth**

# Promise + Events

**Reactive**

# RxJS

## Observable + Iterator



Queue:  
`_requestQueue: Map<TRequestId, Context>`

Zookeeper  
`_agentMap: Map<TDubboInterface, Array<TAgentHostPort>>`  
`_providerMap: Map<TDubboInterface, Array<IProviderProps>>`

Scheduler主要调度:

1. 等待Zookeeper初始化完成，主要等待\_agentSet和\_providerMap
2. 根据\_agentSet创建ServerAgent对象，根据ServerAgent创建SocketPool，SocketPool创建SocketWoker
3. 当某SocketWorker连接成功，上报{pid, host, port}，根据queue队列中的\_requestQueue，寻找么有调度的任务，根据{dubboInterface, version, group}去在\_providerMap中查询可以调用的agentHost列表，根据列表随机选择一个socketAgent，socketAgent随机选择一个SocketWorker
4. 收到onClose事件，查询\_invokeQueue,的pid和requestId，然后从\_scheduleQueue中找到该任务，直接reject返回。
5. onData，根据requestId去\_scheduleQueue找寻resolve直接成功数据返回



比完美更重要的是完成

```
npm install dubbo2.js@1.0.0
```

**3ks @all**

**让晓东给我们插上翅膀,带我们\*\*, 带我们飞 ~**

**Q & A**