Dubbo2.js

from zero to one

胡锋@斑马电商云

Takahashi Method

高橋流簡報法

很大

很多

很快

认真，老司机不等人

Fronted-End development

History

JSP纯真年代


Java


IE BROWSER
safari
搜狗BROWSER
Google Chrome
Opera
世界之窗
FireFox
Netscape
搜狗BROWSER


python

- 前端专注于内容的展示
- 解决浏览器的兼容性

Ajax time

忽如一夜春风来

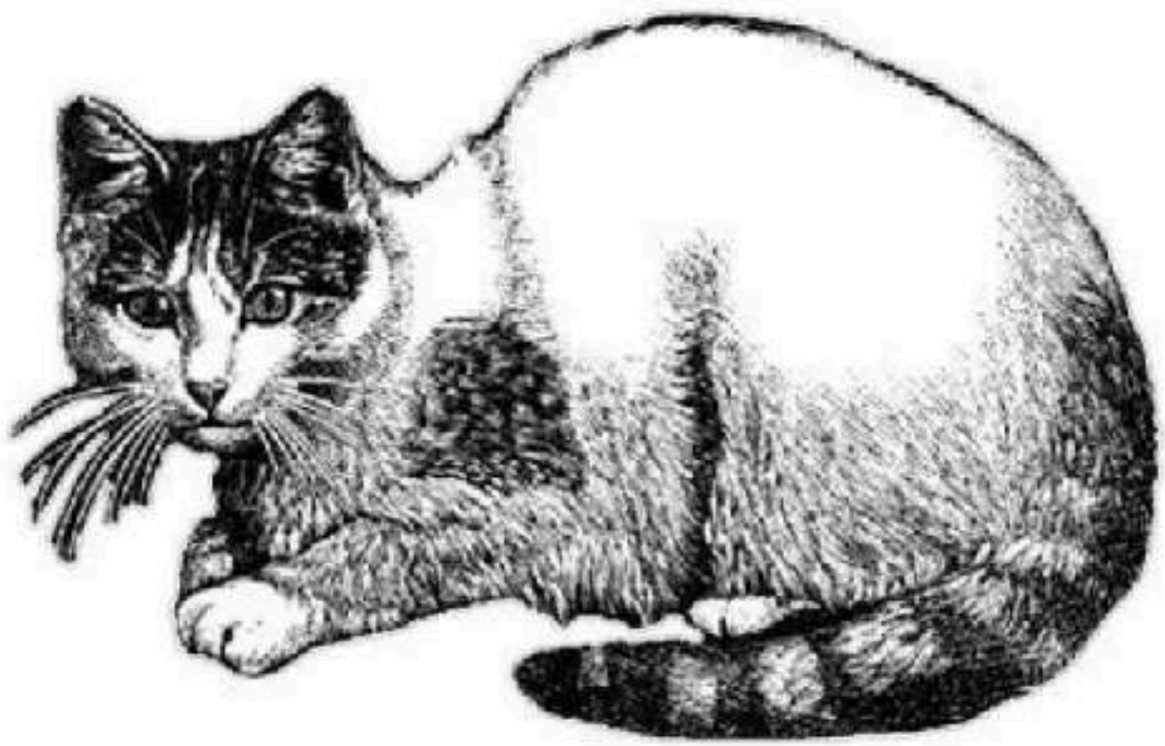
- 模块化
- 组件化
- 工程化

神灯前端系列

最新版

前端工程师
永生指南

走向永生之路



神灯出版社

神灯 著
マジックランプ 訳

challenge

业务的移动化和多端化

广告机

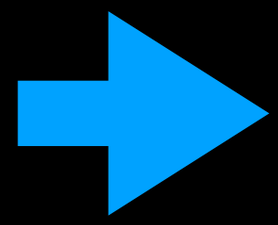
微信公众号　Mobile-web

各种小程序　**多端**　Web

IoT设备

Server　iOS　Android　TV

自动售货机



产品

BI

数据　GA

设计

黑客增长

Javascript

HTML　Typescript　CSS

Java　**语言**　Kotlin

Swift　Objective-c

ReasonML

敏捷交付

调性

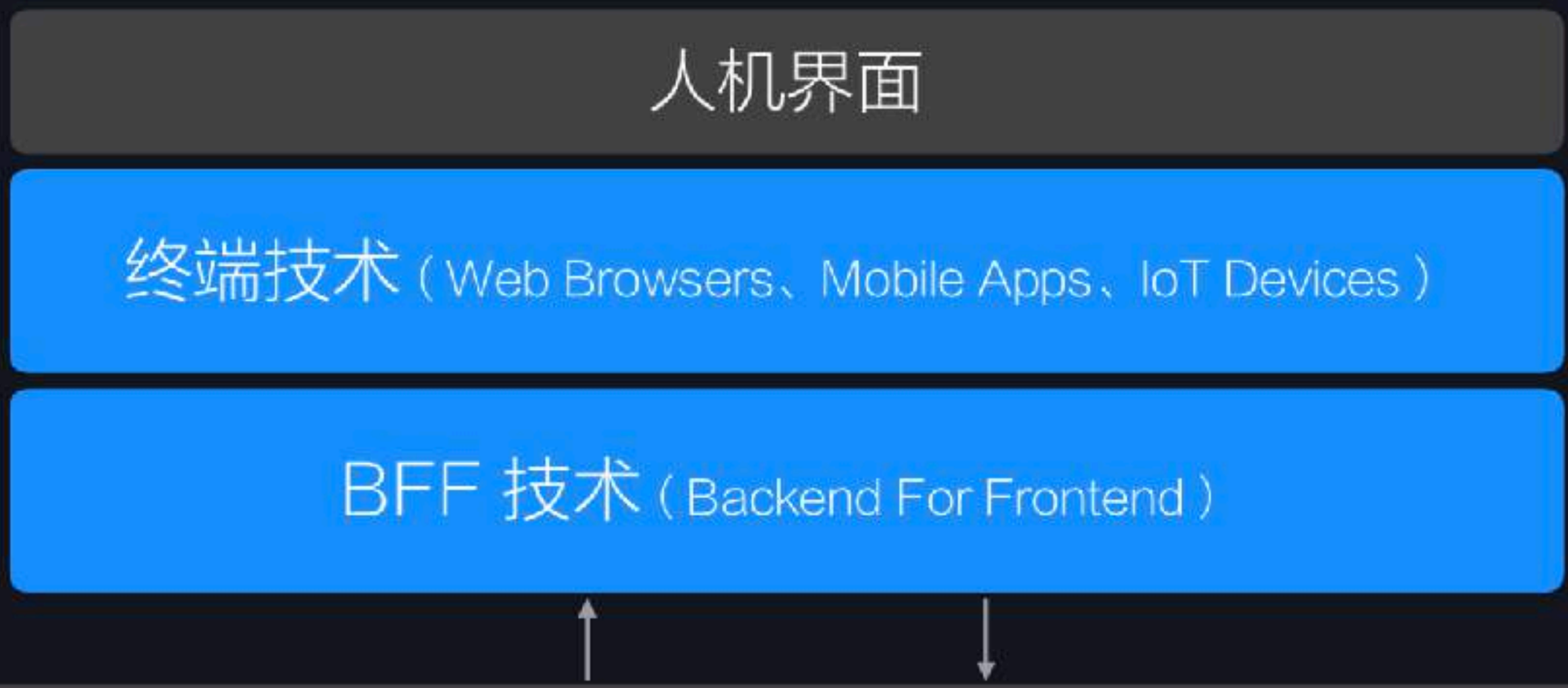
API data tier

- 要求格式灵活多变
- 精确字段
- 敏捷上线
- 历史版本的兼容
- 团队沟通成本
- 比CRUD还缺幸福感

怎么办?

谁难受，谁推动

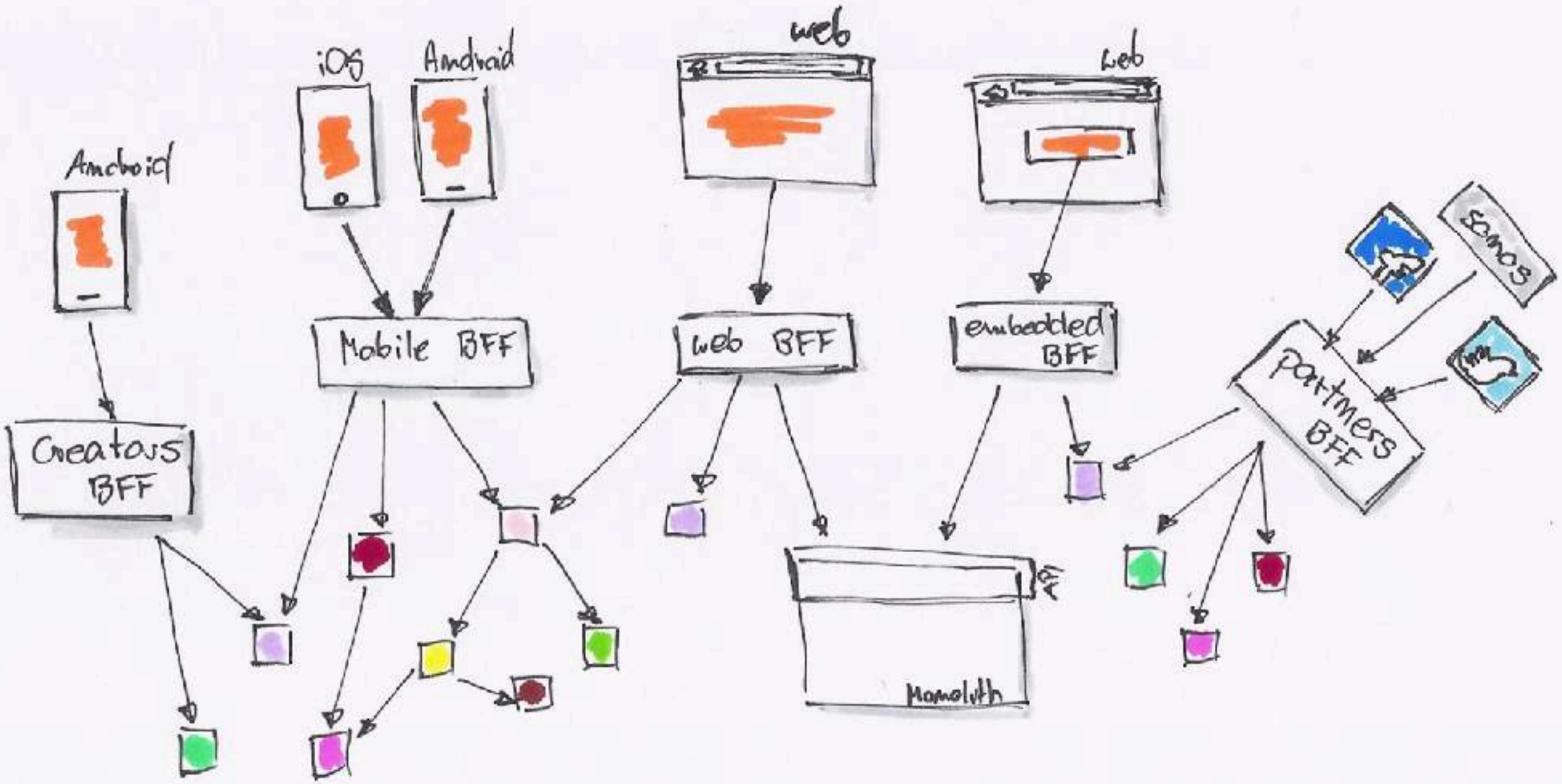
前端技术是后端服务与人机界面的连接器



来自See conf

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专为前端服务，要能够响应前端的变化


Android
Creators BFF
iOS
Android
Mobile BFF
web
web BFF
web
embedded BFF
Monolith
Partners BFF

前端痛点？

怎么成为后端!

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。”

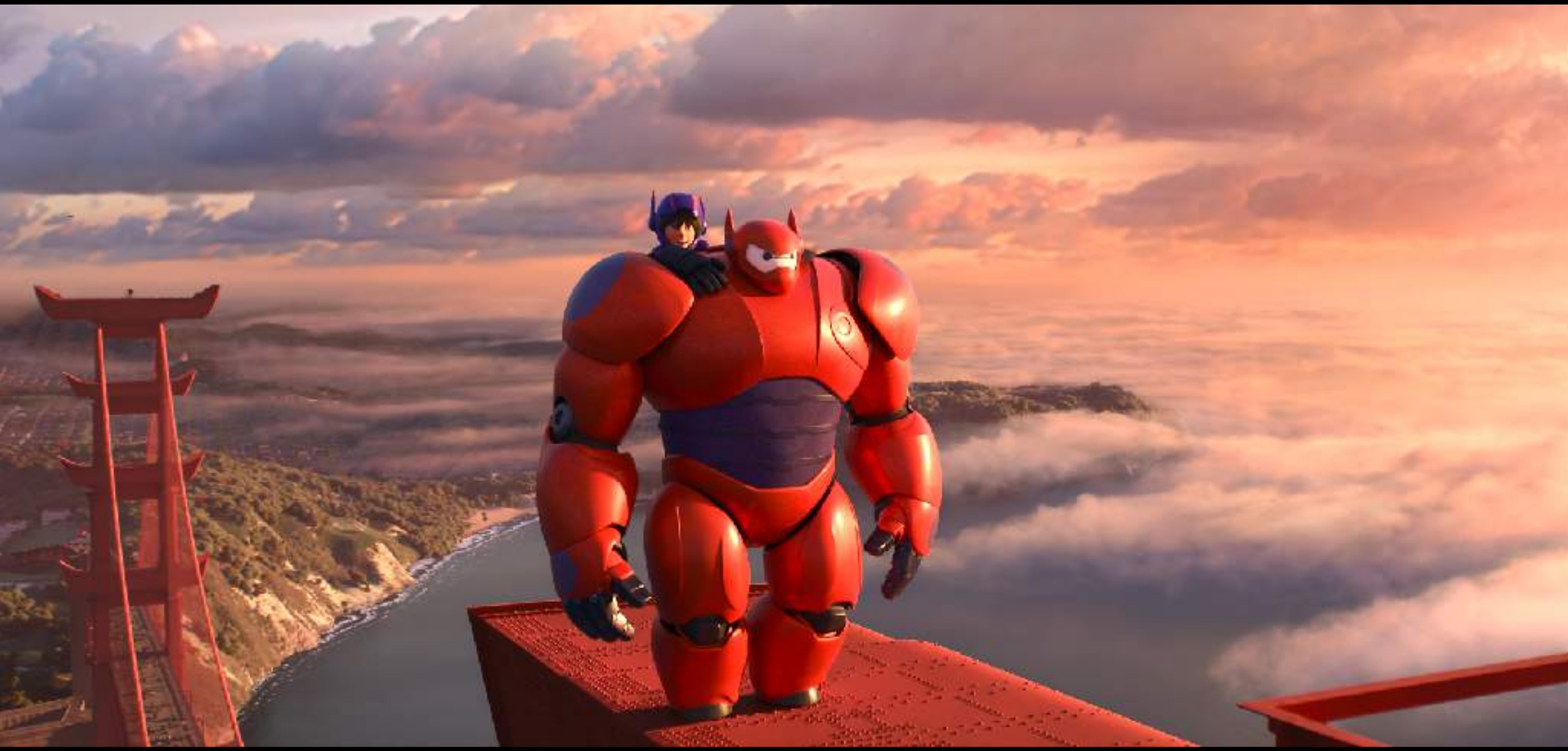
–*Atwood*

Node.JS BFF

小，快，灵

👏Agile Service Team👏

超能陆战队



現実

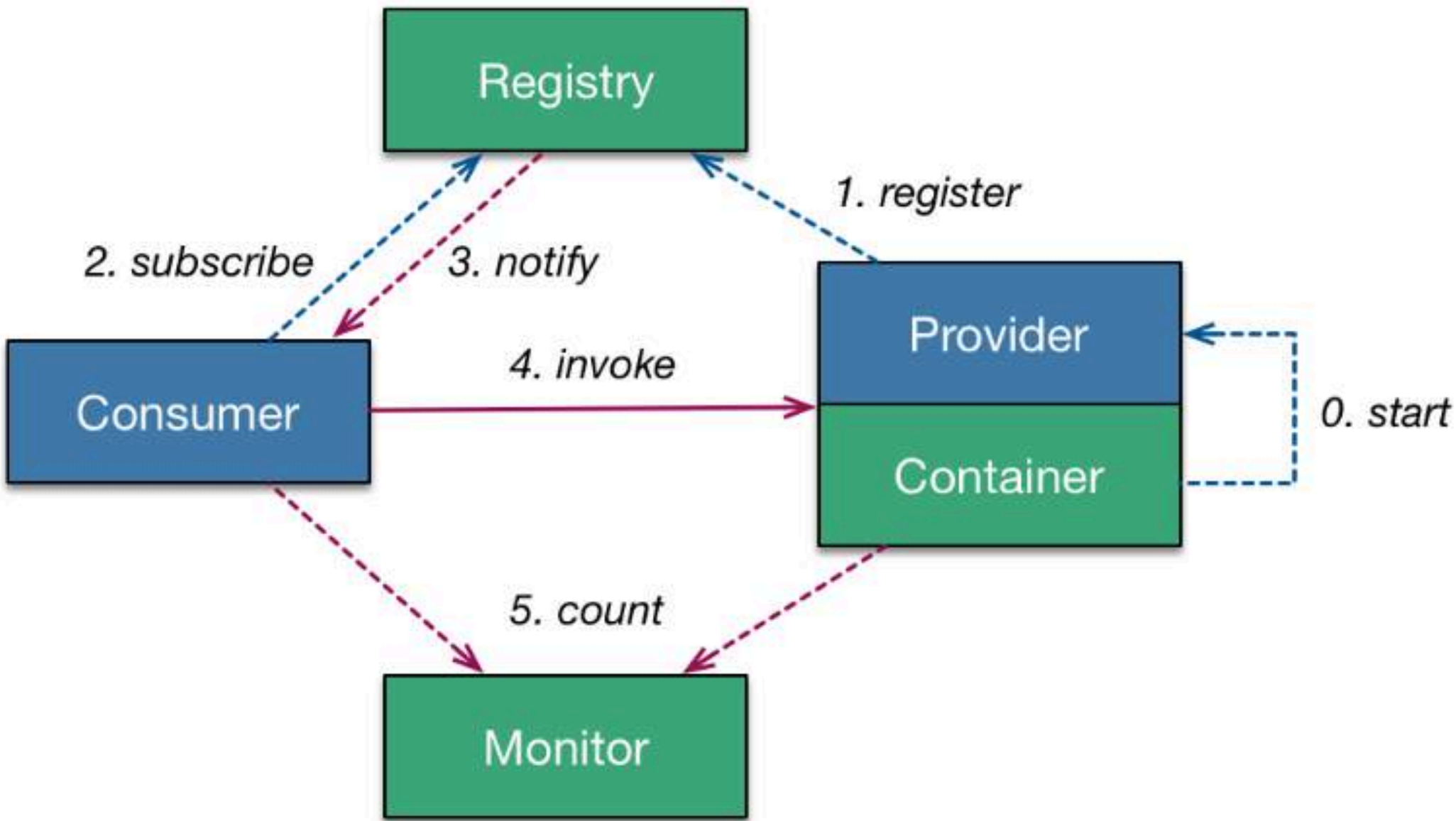
Apache Dubbo (incubating)

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
- 调用不能很透明，开发体验不够友好

念念不忘

必有回响

–《一代宗师》

node connect dubbo with native protocol

梦想在望 脚踏实地

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

//todo-学习

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, protobuff, ….)
- IO(block IO, No-block IO(epoll), async io)

IO?

Node的是什么类型IO?

Serializable

Hessian

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

```
top         ::= value

            # 8-bit binary data split into 64k chunks
binary      ::= 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 true/false
boolean     ::= 'T'
            ::= 'F'

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

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

            # 64-bit IEEE double
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

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

            # list/vector
list        ::= 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

            # 64-bit signed long integer
long        ::= '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 pai
            ::= 'H' (value value)* 'Z'       # untyped key, value

            # null value
null        ::= 'N'

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

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

            # UTF-8 encoded character string split into 64k chunk
string      ::= 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

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

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

Bool Int Date Null

Nested long

2015 D2

node-modules / hessian.js

Watch 15 | Unstar 90 | Fork 21

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

😆😆😆😆

❤️❤️❤️

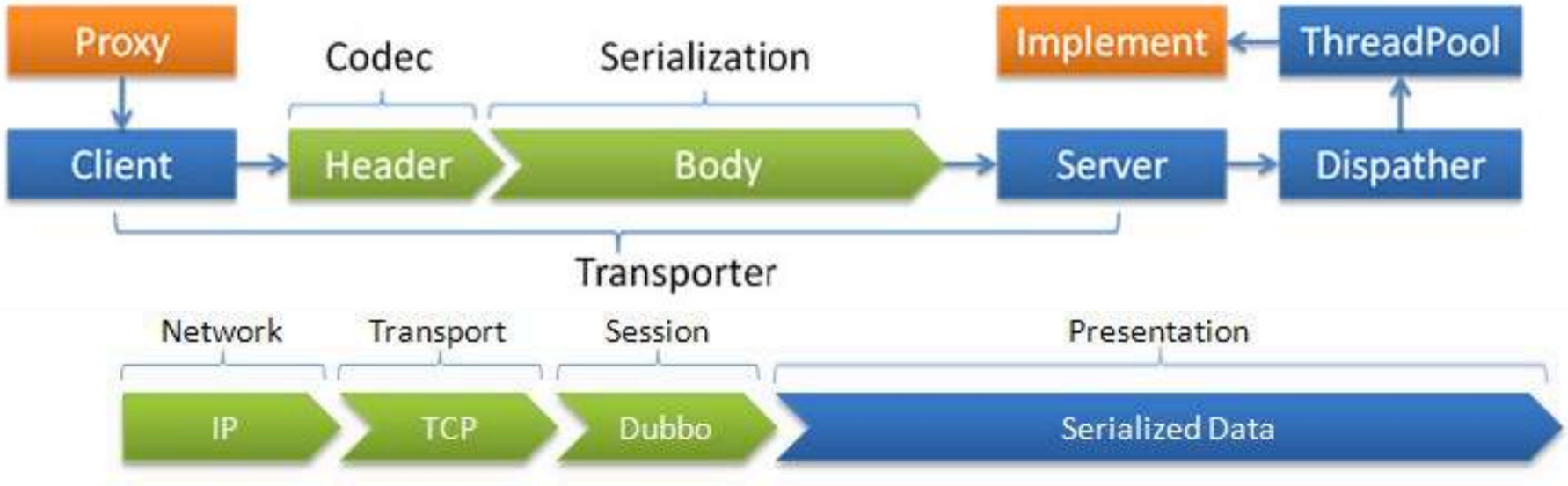
🚀🚀🚀

6666

厉害了:)

👏👏👏

Dubbo Protocol



IP

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)					

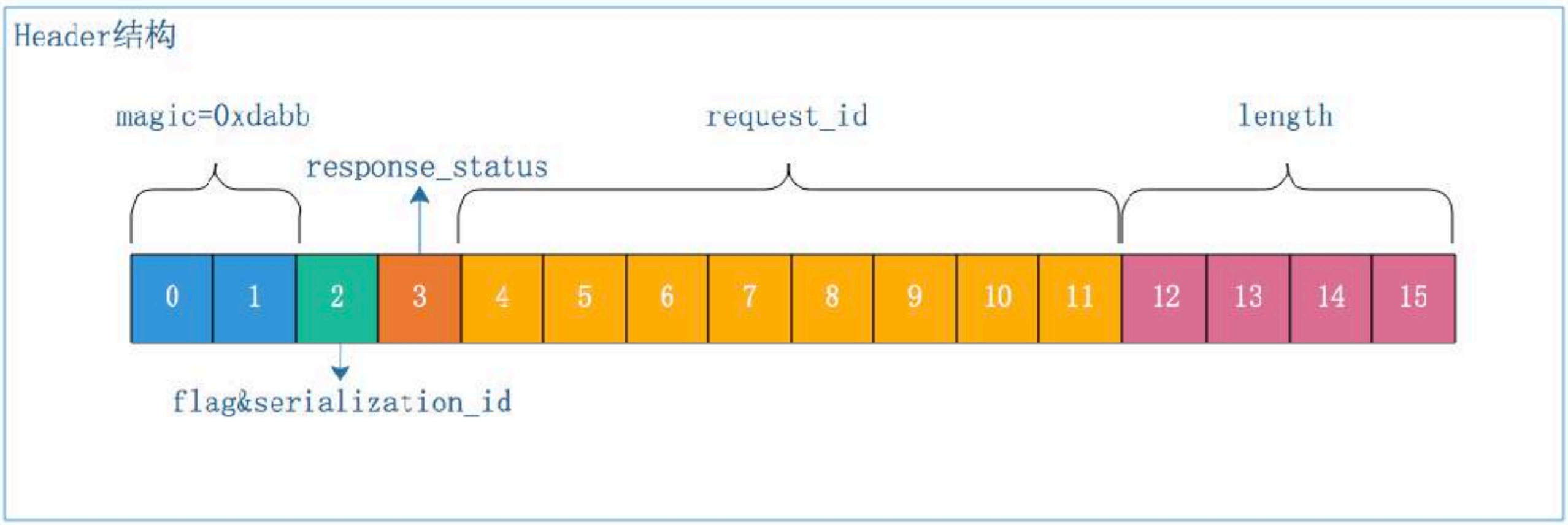
TCP

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	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size	
128	Checksum										Urgent pointer (if URG set)	
160	Options (if Data Offset > 5)											padding

Dubbo

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



Dubbo 报文格式


head
标志位（请求、响应、事件）
status
序列化
request body
版本号
接口路径
接口版本
方法名称
参数类型
参数值
response body
结果标志(无结果、有结果、异常)
结果

Hack: baby step

```
//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_REQEUST | 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);
    }
  }
```

```java
/**
 * 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, ObjectOutput out, Object data) throws IOE
        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);
        }
    }
}
```

```
//com.alibaba.dubbo.remoting.exchange.codec.ExchangeCodec.encodeResponse/decode
export function decode<T>(bytes: Buffer): IDubboResponse<T> {
  let res = null;
  let err = null;

  // set request and serialization flag.
  const requestIdBuff = Buffer.alloc(8);
  requestIdBuff[0] = bytes[4];
  requestIdBuff[1] = bytes[5];
  requestIdBuff[2] = bytes[6];
  requestIdBuff[3] = bytes[7];
  requestIdBuff[4] = bytes[8];
  requestIdBuff[5] = bytes[9];
  requestIdBuff[6] = bytes[10];
  requestIdBuff[7] = bytes[11];

  const requestId = convertBinaryNum(requestIdBuff, 8);
  log(`decode parse requestId: ${requestId}`);

  // const typeId = bytes[2];

  // get response status.
  const status = bytes[3];

  log(···
  );

  if (status != DUBBO_RESPONSE_STATUS.OK) {···
  }

  //com.alibaba.dubbo.rpc.protocol.dubbo.DecodeableRpcResult
  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

Only Serializable

Move fast break things

工程化

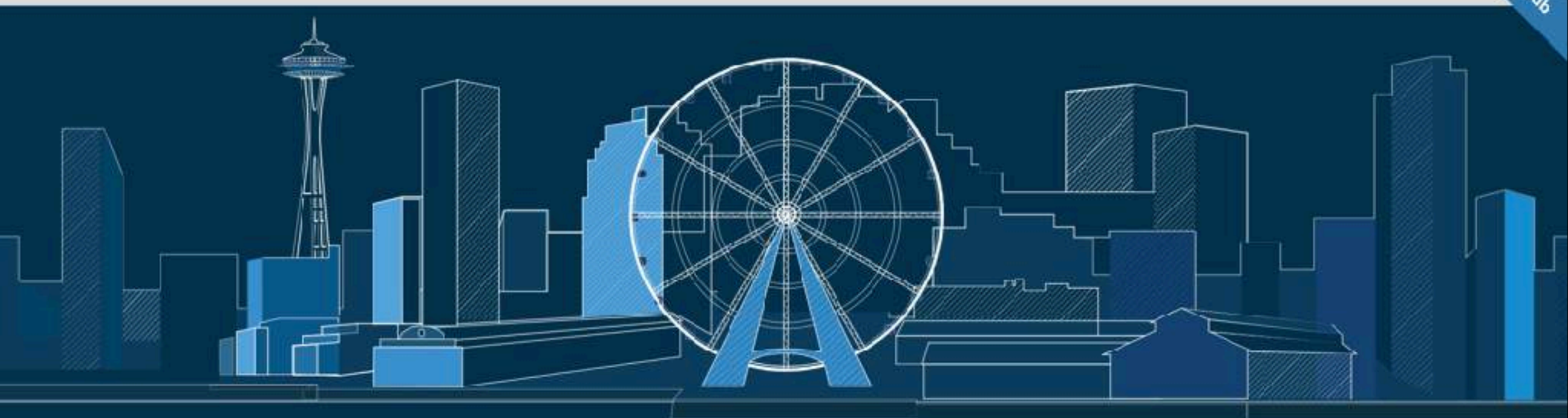
Mono-Repo

Lerna

A tool for managing JavaScript projects with multiple packages.

```
~/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

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

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

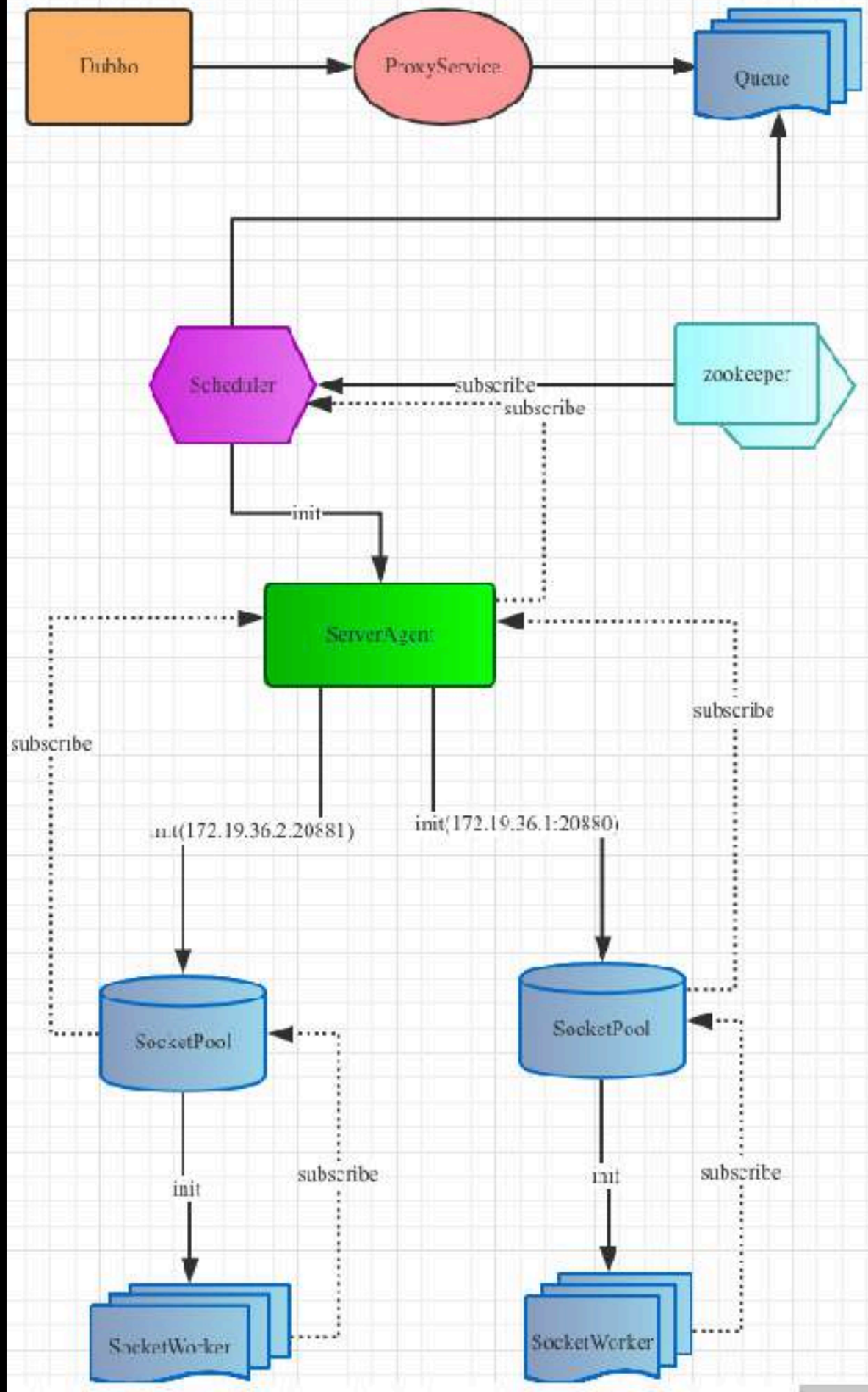
Fast interactive watch mode runs only test files related to changed files and is optimized to give signal quickly.

Snapshot Testing

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

BBB

理清职责
划清边界
识别聚合根



Queue:
_requestQueue: Map<IRequestId, 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,某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

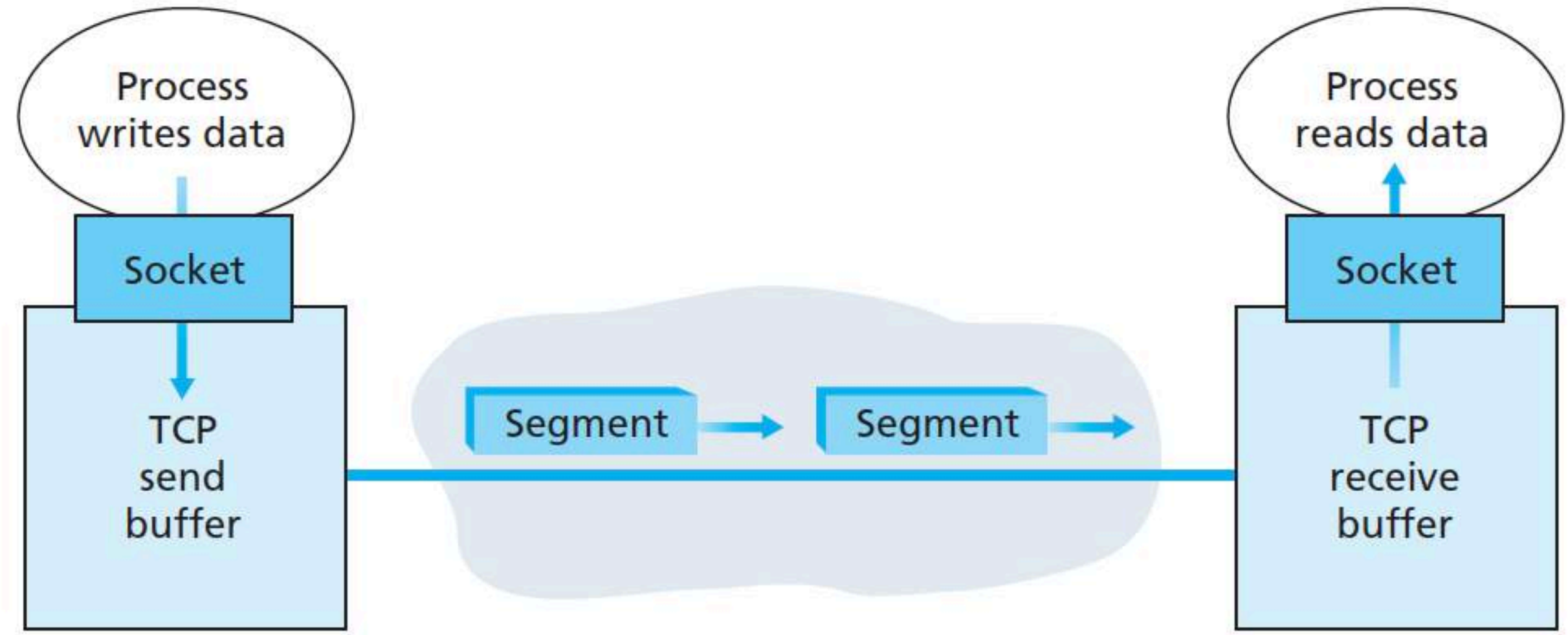
Asynchronous?

//Queue大法真心好用

Big Bug 😭 嫩

ab -n 10000 -c 100

TCP缺包、粘包、顺序


Process writes data
Socket
TCP send buffer
Segment
Segment
Process reads data
Socket
TCP receive buffer

Buffer-decode

疯狂基建

ELK Logger

Kafka-client

Egg-qm-logger

Egg-log-tracer

Tracing

Request -> uuid -> ? -> ?

Dubbo Invoke 我在哪?

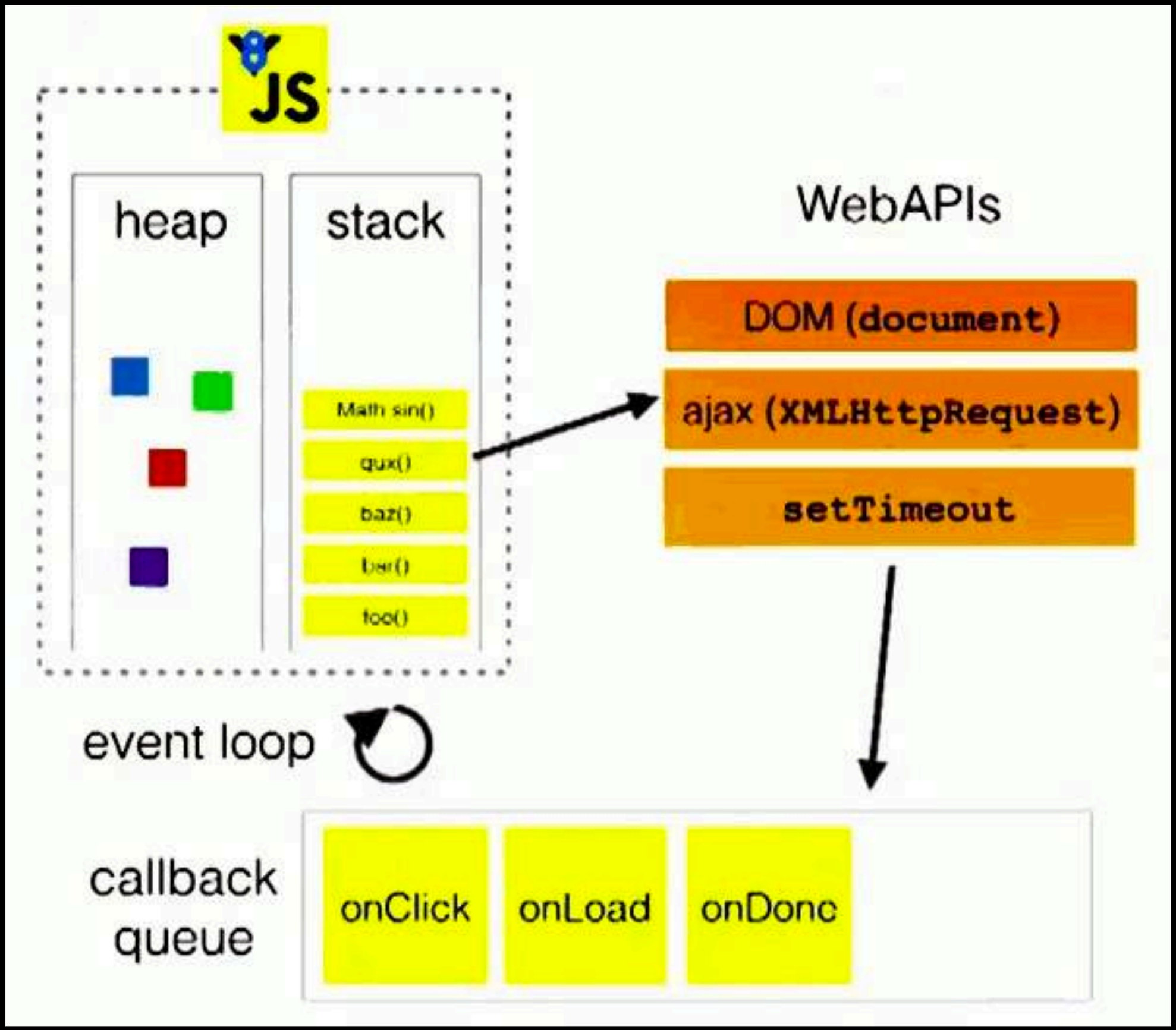
Explicit

vs

Implicit

Java

ThreadLocal


JS
heap
stack
Math.sin()
qux()
baz()
bar()
foo()
WebAPIs
DOM (document)
ajax (XMLHttpRequest)
setTimeout
event loop
callback queue
onClick
onLoad
onDone

Zone.js

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:

等一等

async/await....


救星!
S' SAVIOUR

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().
//
```

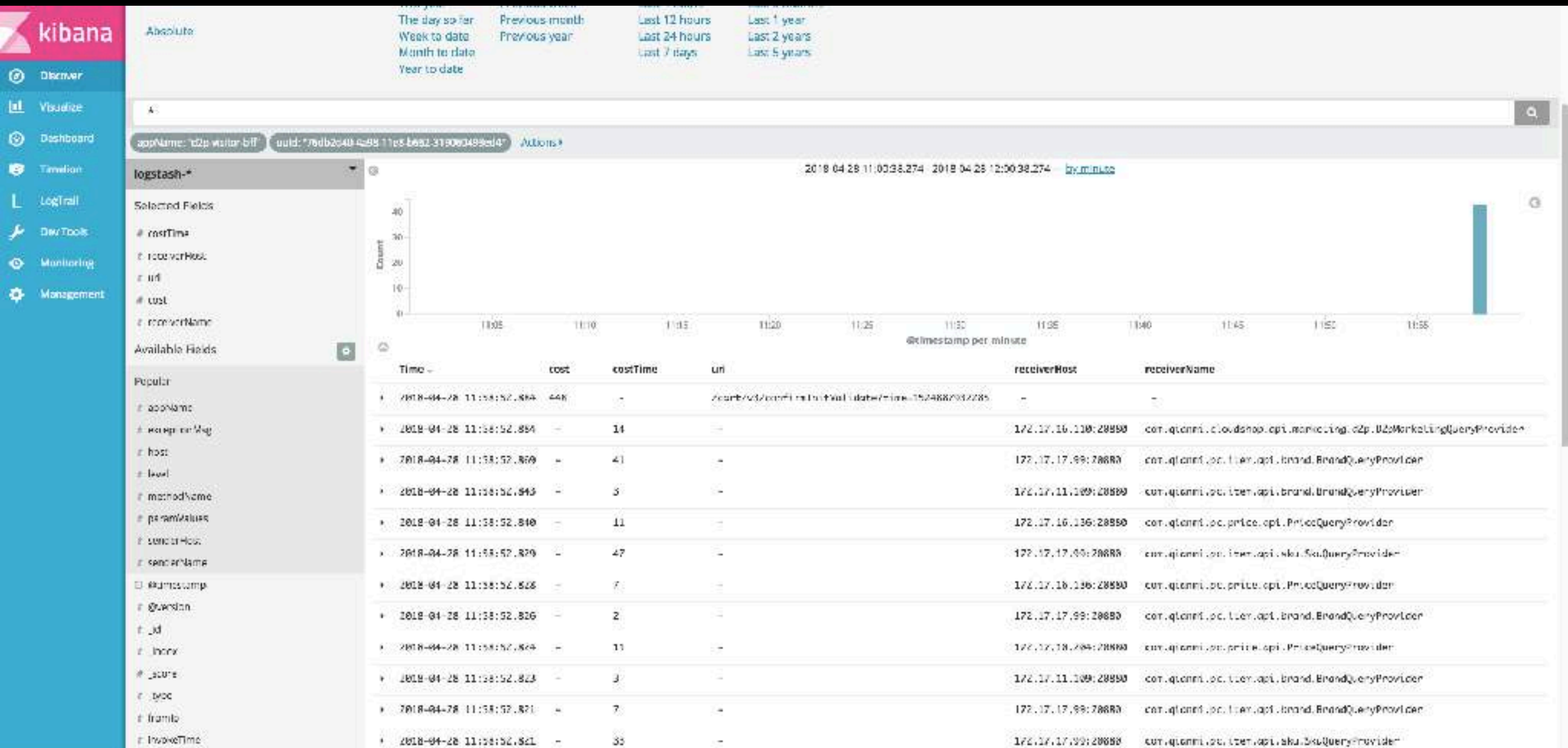
```ts
import async_hooks from 'async_hooks';
import debug from 'debug';
const log = debug('dubbo:zone');

//alias type
export type AsyncId = number;
export type RootAsyncId = number;

/**
 * ZoneContext 期待Zone的规范早日落地
 */
export class ZoneContext {
  constructor() {
    log('init ZoneContext');
    this.rootMap = new Map();
    this.statckFrameMap = new Map();
    this.initAsyncHook();
  }

  private rootMap: Map<AsyncId, object>;
  private statckFrameMap: Map<AsyncId, RootAsyncId>;
```

npm install zone-context


kibana
Discover
Visualize
Dashboard
Timelion
LogTrail
Dev Tools
Monitoring
Management
Absolute
The day so far
Week to date
Month to date
Year to date
Previous month
Previous year
Last 12 hours
Last 24 hours
Last 7 days
Last 1 year
Last 2 years
Last 5 years
Actions ▸
logstash-*
Selected Fields
costTime
receiverHost
url
cost
receiverName
Available Fields
Popular
appName
host
level
methodName
paramValues
senderHost
senderName
@timestamp
@version
_id
_index
_score
_type
InvokeTime
Count
@timestamp per minute
Time
cost
costTime
url
receiverHost
receiverName

Extention

Request response chain

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 ❤️

比完美更重要的是完成

只有完成才有机会完美

npm install dubbo2.js

dubbo2.js

Nodejs 💕 Dubbo RPC Service

GET STARTED LEARN MORE

Keep it Simple

build tools for Humans. 💗

Scale & Performance

Nodejs no-blocking io and cluster 🚀

Easy to Extend

Write plugin is so easy 👏

Who's Using This?

但是那个调用不透明的问题呢?

Y.. Ye.. Yes...

你真的是一个认真的码神!

👏👏👏

3ks ❤️ @all