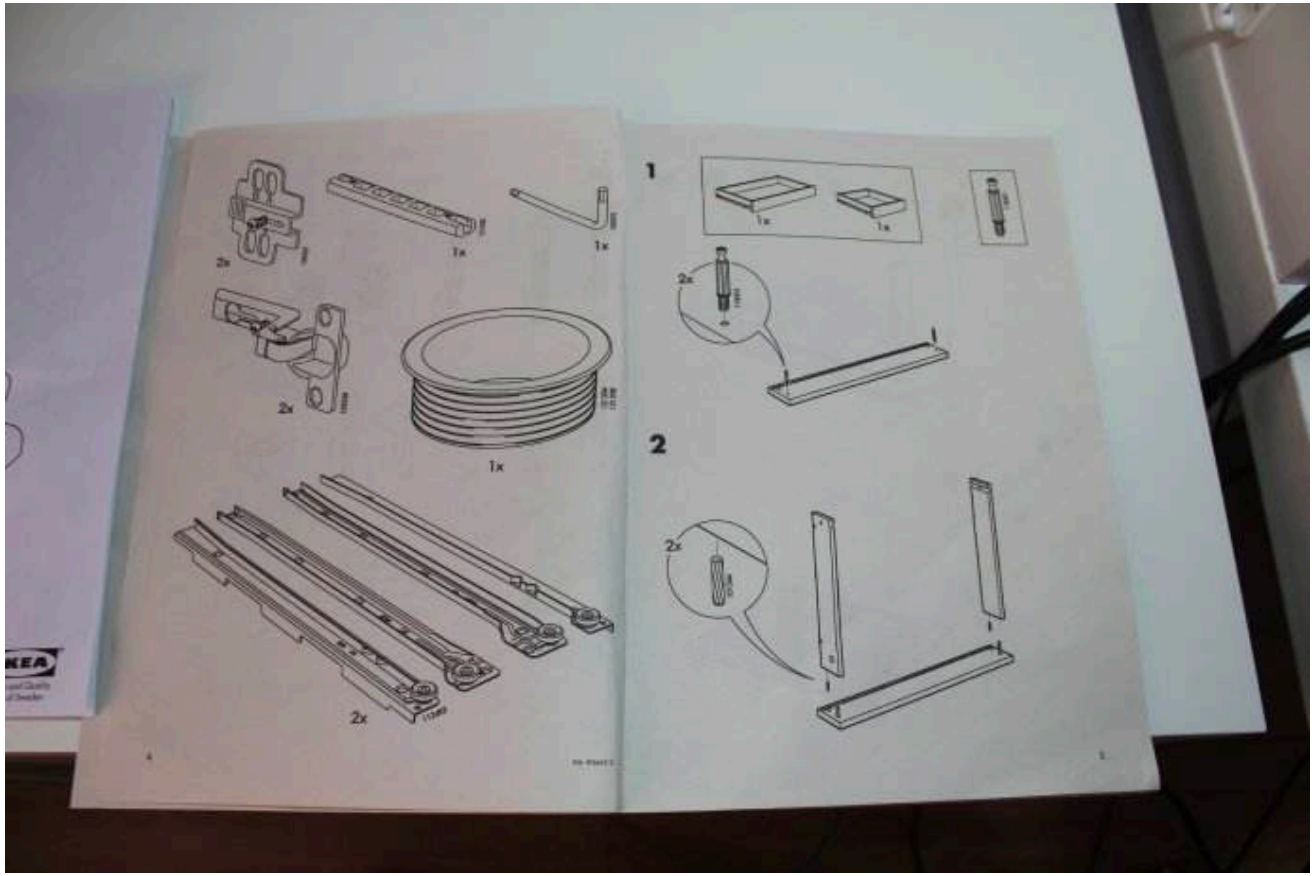
什么是技术预研？

极客时间

扫码试看/订阅

《Node.js 开发实战》视频课程

什么是技术预研

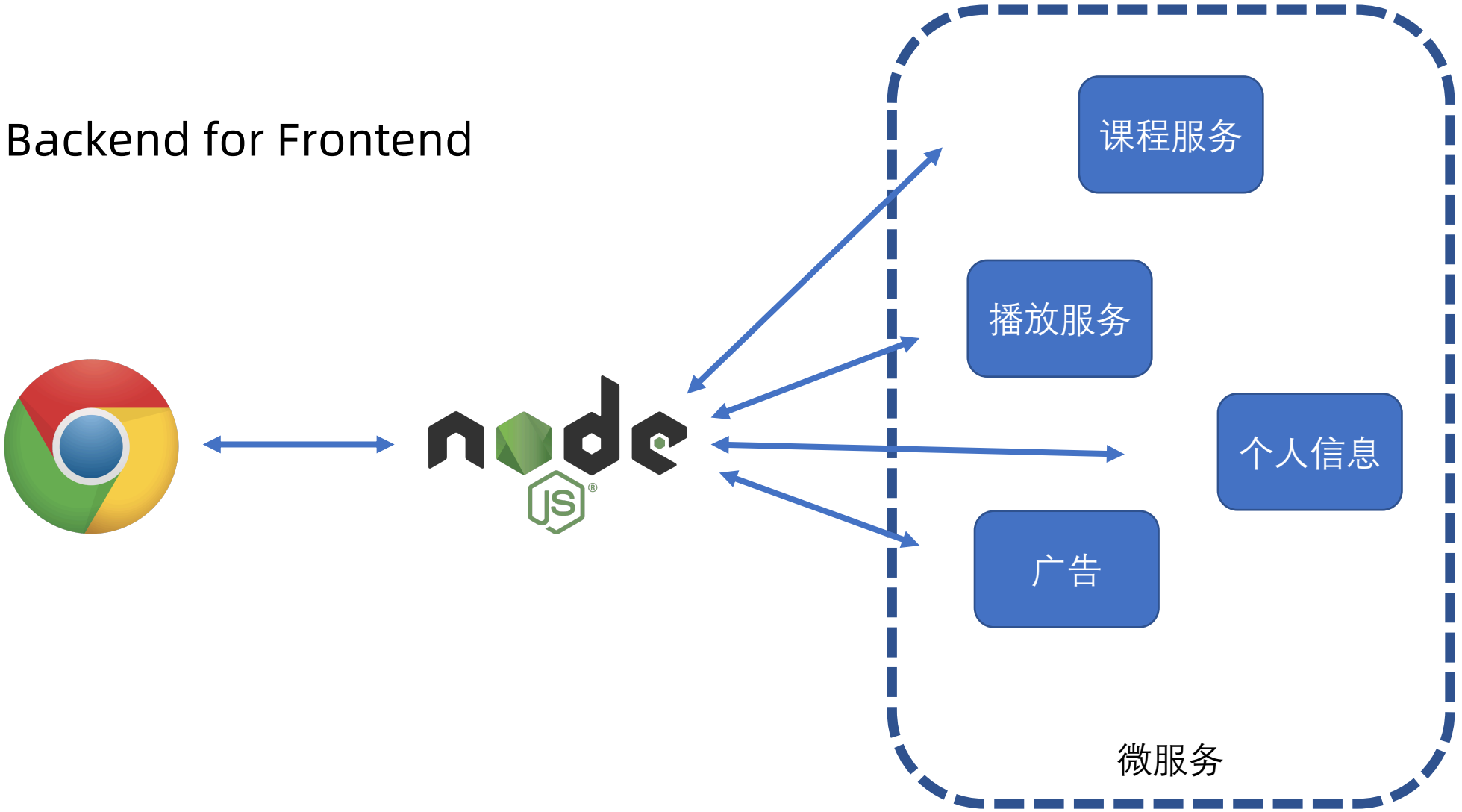


什么是技术预研

- 分析要做的需求，找出技术难点。
- 针对每个技术难点设计 demo 进行攻克。

BFF 层

- Backend for Frontend



BFF 层

- 对用户侧提供 HTTP 服务
- 使用后端 RPC 服务

BFF 层

- 对用户侧提供 HTTP 服务
- 使用后端 RPC 服务

- 在这之前，需要了解 Node.js 怎么跑起来。

Node.js 开发环境安装

Chrome 的安装

Visual Studio Code 的安装

Node.js 的安装

Node.js 第一个实战 - 石头剪刀布游戏

Node.js 第一个实战 - 石头剪刀布游戏

- 运行方式
- Node.js 全局变量

CommonJS 模块规范

<script />

<script />

- 脚本变多时，需要手动管理加载顺序。

<script />

- 脚本变多时，需要手动管理加载顺序。
- 不同脚本之间逻辑调用，需要通过全局变量的方式。

<script />

- 脚本变多时，需要手动管理加载顺序。
- 不同脚本之间逻辑调用，需要通过全局变量的方式。
- 没有 html 怎么办？

CommonJS 模块规范

- JavaScript 社区发起，在 Node.js 上应用并推广。
- 后续也影响到了浏览器端 JavaScript。

npm 及 npm 包

npm

- npm 是什么
  - Node.js 的包管理工具

npm

- npm 是什么
  - Node.js 的包管理工具

- 包是什么
  - 别人写的 Node.js 模块

npm

- npm 上的著名大神
  - TJ Holowaychunk
  - Mafintosh
  - Dominictarr
  - ……

npm

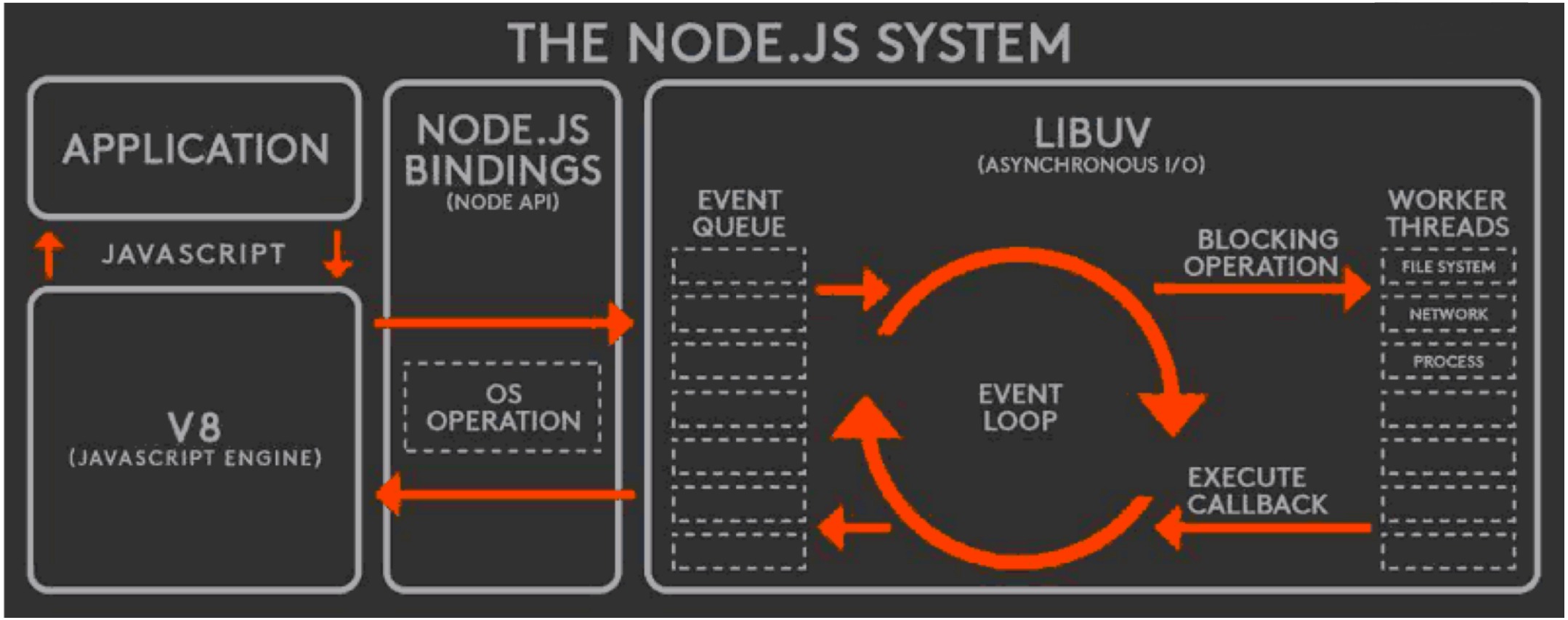
- npm event-stream 事件

npm

- 没有 npm，也不会有现在这么繁荣的 JS 社区。

Node.js 内置模块

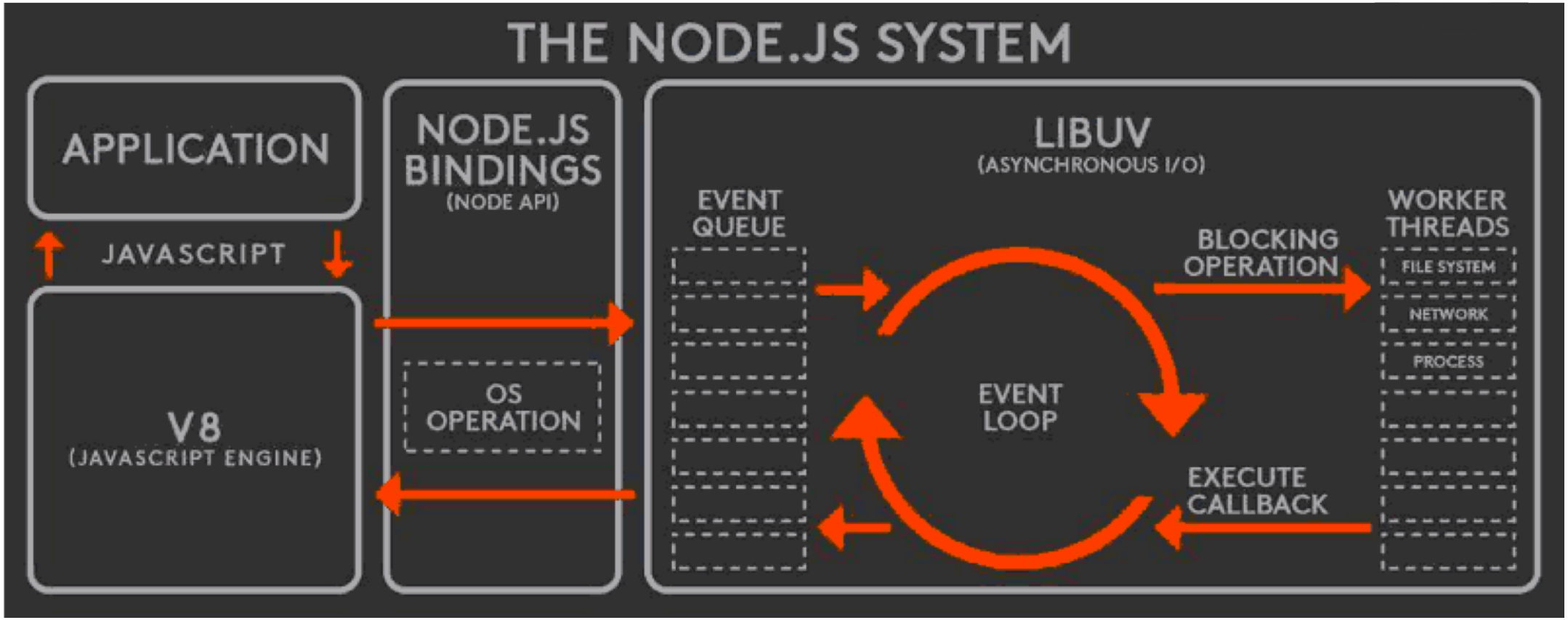
Node.js 内置模块


THE NODE.JS SYSTEM
APPLICATION
JAVASCRIPT
V8
(JAVASCRIPT ENGINE)
NODE.JS
BINDINGS
(NODE API)
OS
OPERATION
LIBUV
(ASYNCHRONOUS I/O)
EVENT
QUEUE
EVENT
LOOP
BLOCKING
OPERATION
EXECUTE
CALLBACK
WORKER
THREADS
FILE SYSTEM
NETWORK
PROCESS

Node.js 内置模块

- OS 模块

Node.js 内置模块


THE NODE.JS SYSTEM
APPLICATION
JAVASCRIPT
V8
(JAVASCRIPT ENGINE)
NODE.JS
BINDINGS
(NODE API)
OS
OPERATION
LIBUV
(ASYNCHRONOUS I/O)
EVENT
QUEUE
EVENT
LOOP
BLOCKING
OPERATION
EXECUTE
CALLBACK
WORKER
THREADS
FILE SYSTEM
NETWORK
PROCESS

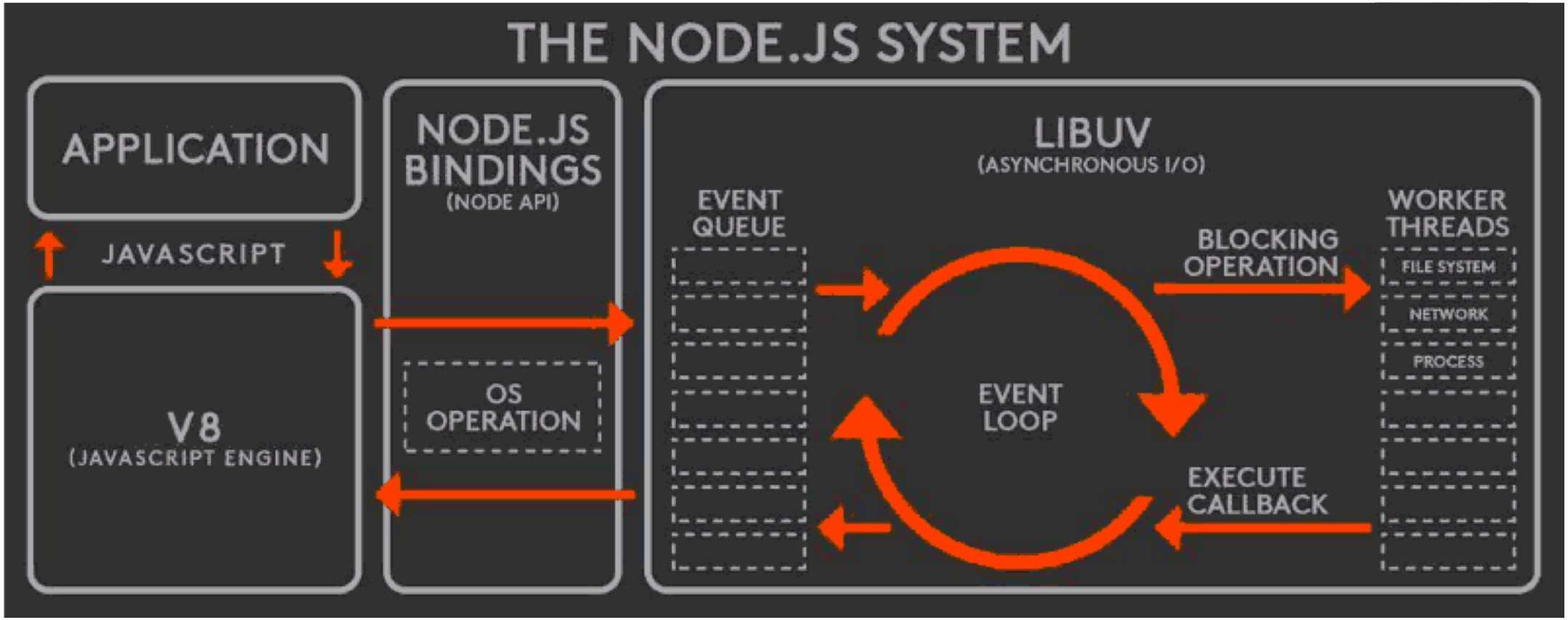
Node.js 内置模块

- EventEmitter
  - 观察者模式
    - addEventListener
    - removeEventListener

Node.js 内置模块

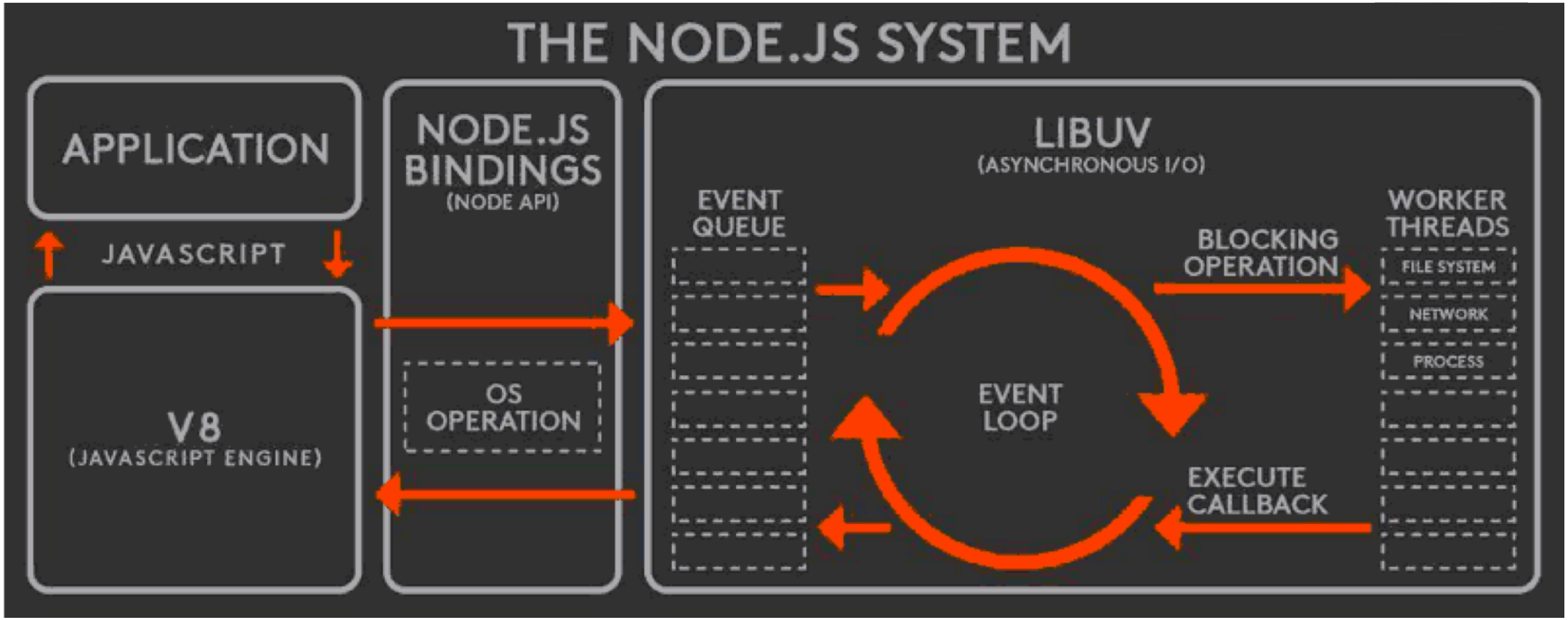
- EventEmitter
  - 观察者模式
  - 调用 vs 抛事件
    - 关键在于“不知道被通知者存在”
    - 以及“没有人听还能继续下去”

Node.js 内置模块


THE NODE.JS SYSTEM
APPLICATION
JAVASCRIPT
V8
(JAVASCRIPT ENGINE)
NODE.JS
BINDINGS
(NODE API)
OS
OPERATION
LIBUV
(ASYNCHRONOUS I/O)
EVENT
QUEUE
EVENT
LOOP
BLOCKING
OPERATION
EXECUTE
CALLBACK
WORKER
THREADS
FILE SYSTEM
NETWORK
PROCESS

Node.js 异步

Node.js 的异步


THE NODE.JS SYSTEM
APPLICATION
JAVASCRIPT
V8
(JAVASCRIPT ENGINE)
NODE.JS
BINDINGS
(NODE API)
OS
OPERATION
LIBUV
(ASYNCHRONOUS I/O)
EVENT
QUEUE
EVENT
LOOP
BLOCKING
OPERATION
EXECUTE
CALLBACK
WORKER
THREADS
FILE SYSTEM
NETWORK
PROCESS

Node.js 的非阻塞 I/O

Node.js 的非阻塞 I/O

- I/O 即 Input/Output，一个系统的输入和输出。
- 阻塞 I/O 和非阻塞 I/O 的区别就在于系统接收输入再到输出期间，能不能接收其他输入。

Node.js 的非阻塞 I/O

- 吃饭

Node.js 的非阻塞 I/O

- 去饭堂吃：排队打饭

- 出去吃：餐厅点菜

Node.js 的非阻塞 I/O

- 排队打饭 vs 餐厅点菜
- 对于点菜人员：
  - 排队打饭是阻塞 I/O
  - 餐厅点菜是非阻塞 I/O

Node.js 的非阻塞 I/O

- I/O 即 Input/Output，一个系统的输入和输出。
- 阻塞 I/O 和非阻塞 I/O 的区别就在于系统接收输入再到输出期间，能不能接收其他输入。

Node.js 的非阻塞 I/O

- I/O 即 Input/Output，一个系统的输入和输出。
- 阻塞 I/O 和非阻塞 I/O 的区别就在于系统接收输入再到输出期间，能不能接收其他输入。

- 系统=食堂阿姨/服务生，输入=点菜，输出=端菜。
- 饭堂阿姨只能一份一份饭地打 -> 阻塞 I/O
- 服务生点完菜之后可以服务其他客人 -> 非阻塞 I/O

Node.js 的非阻塞 I/O

- “这个 Node.js 问题怎么解决？在线等，急。”

Node.js 的非阻塞 I/O

- “这个 Node.js 问题怎么解决？在线等，急。”

- -> 阻塞 I/O

Node.js 的非阻塞 I/O

小芳帮妈妈做家务，需要做：用洗衣机洗衣服（20分钟）、扫地（10分钟）、整理书桌（10分钟）、晾衣服（5分钟）．你能不能设计一个巧妙合理的新顺序，使小芳最少花（　　）分钟可以完成这些事?

A. 20

B. 25

C. 30

D. 35

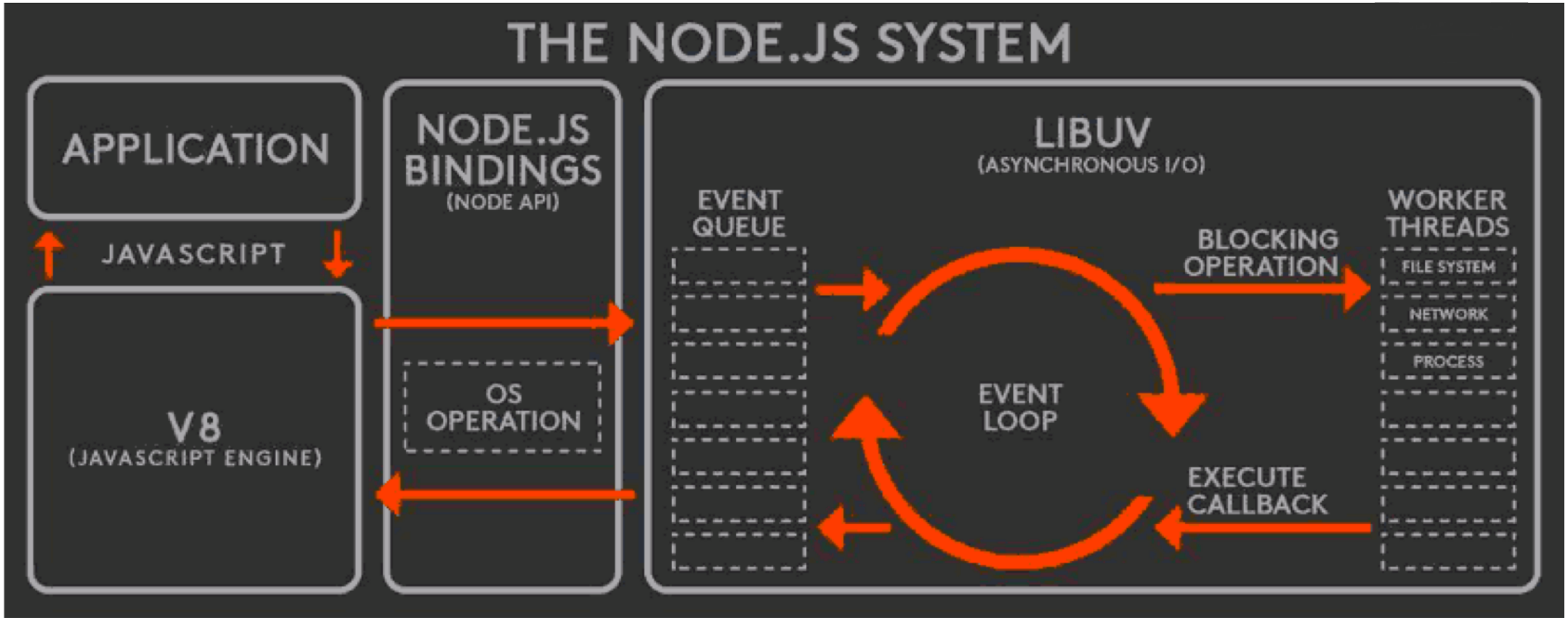
Node.js 的非阻塞 I/O

- 理解非阻塞 I/O 的要点在于
  - 确定一个进行 Input/Output 的系统。
  - 思考在 I/O 过程中，能不能进行其他 I/O。

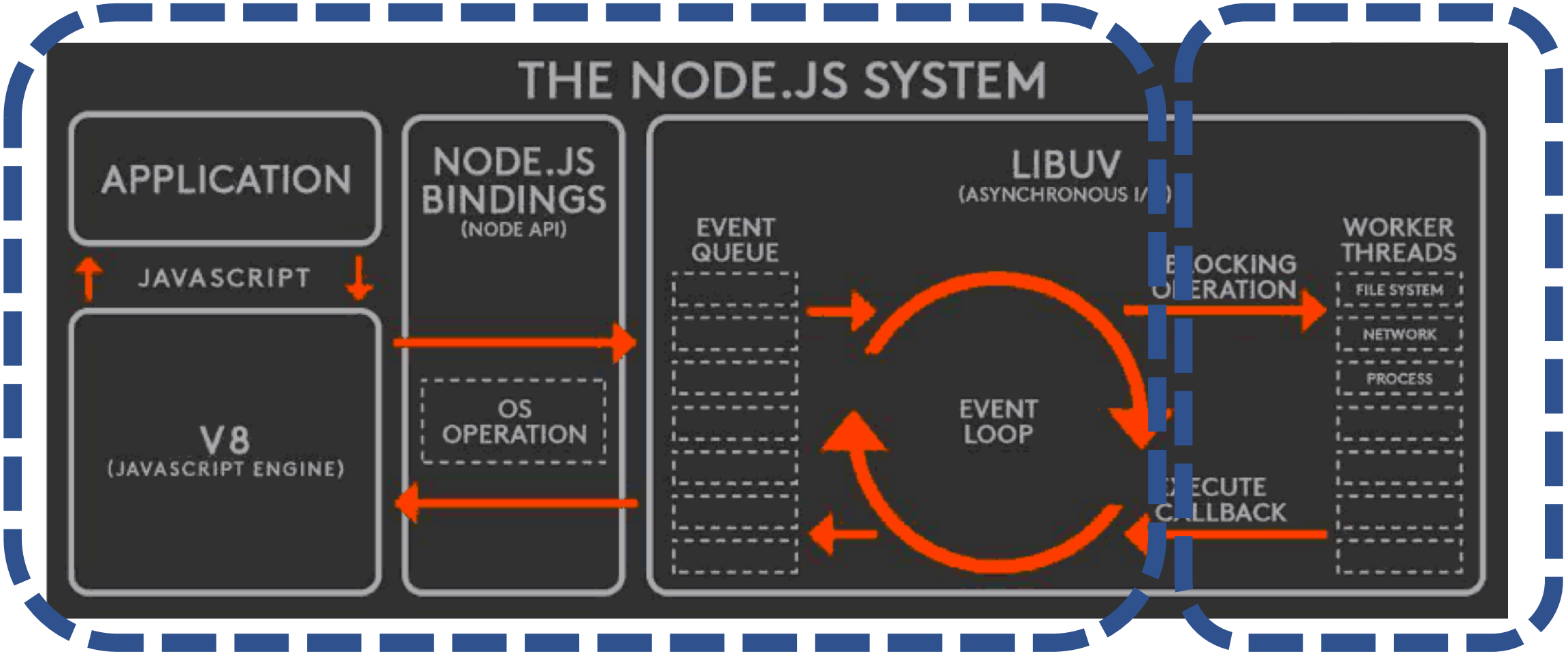
Node.js 的非阻塞 I/O

- 代码演示 （glob）

Node.js 的非阻塞 I/O



Node.js 的非阻塞 I/O



Node.js 线程

其他 c++ 线程

Node.js 异步编程 - callback

Node.js 异步编程 - callback

- 回调函数格式规范
  - error-first callback
  - node-style callback
- 第一个参数是 error，后面的参数才是结果。

Node.js 异步编程 - callback

- 异步流程控制

- npm: async.js

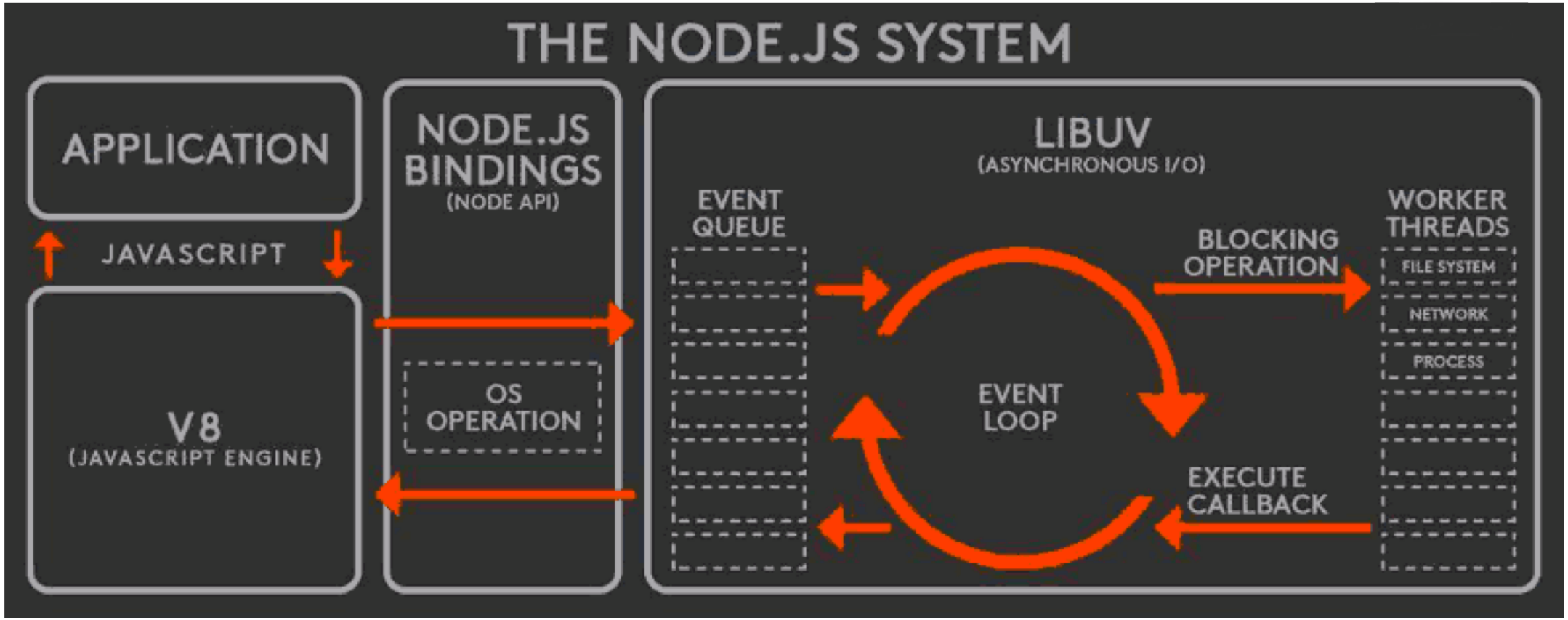
Node.js 异步编程 - callback

- 异步流程控制

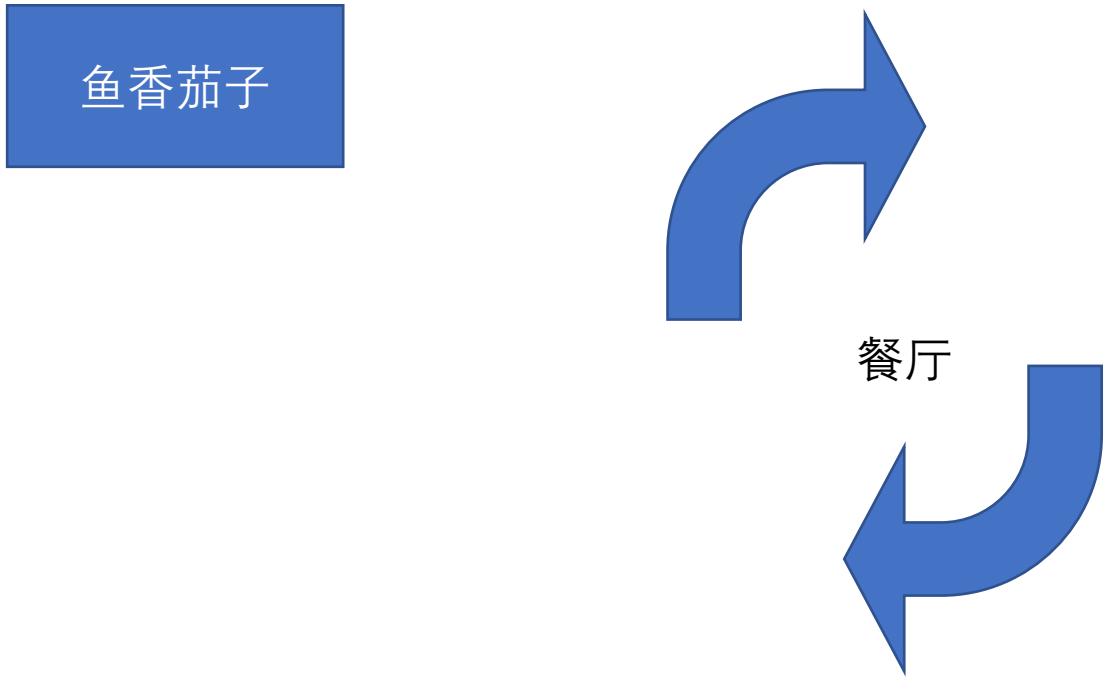
- npm: async.js
- thunk

Node.js 事件循环

事件循环


THE NODE.JS SYSTEM
APPLICATION
JAVASCRIPT
V8
(JAVASCRIPT ENGINE)
NODE.JS
BINDINGS
(NODE API)
OS
OPERATION
LIBUV
(ASYNCHRONOUS I/O)
EVENT
QUEUE
EVENT
LOOP
BLOCKING
OPERATION
WORKER
THREADS
FILE SYSTEM
NETWORK
PROCESS
EXECUTE
CALLBACK

事件循环



事件循环

鱼香茄子

鱼香茄子 线程

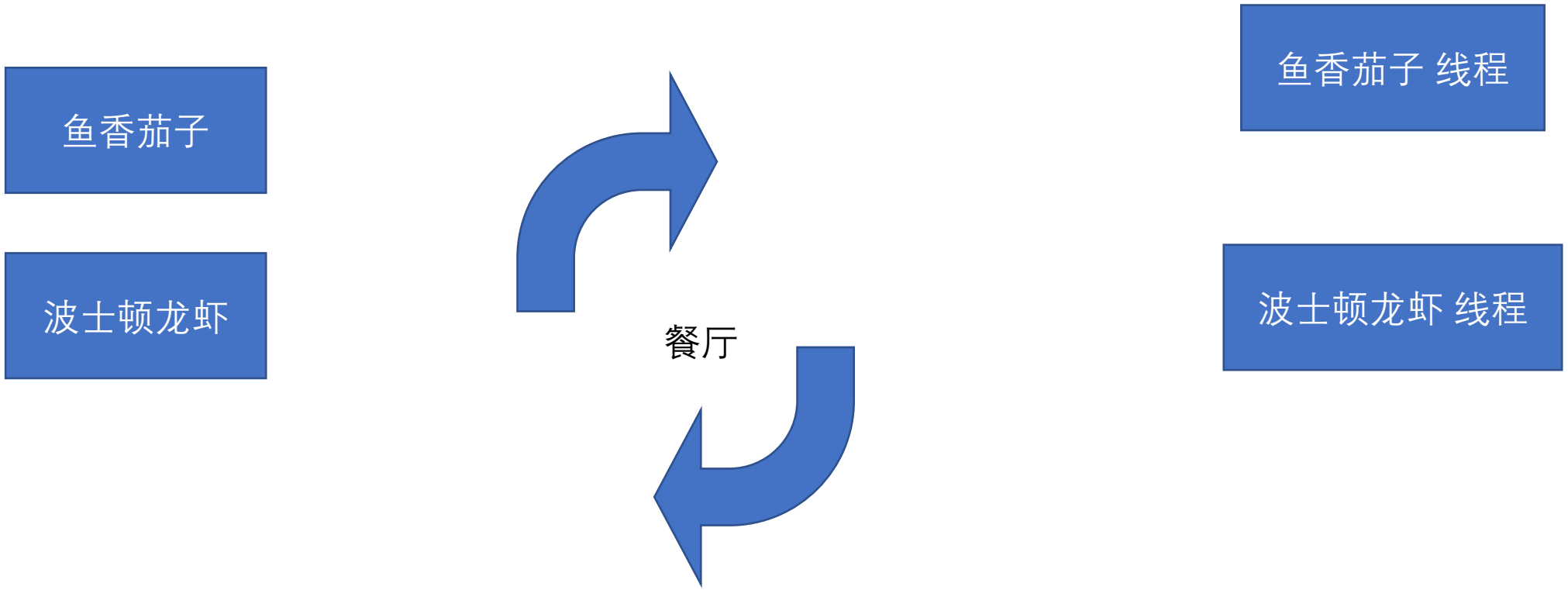
事件循环

鱼香茄子

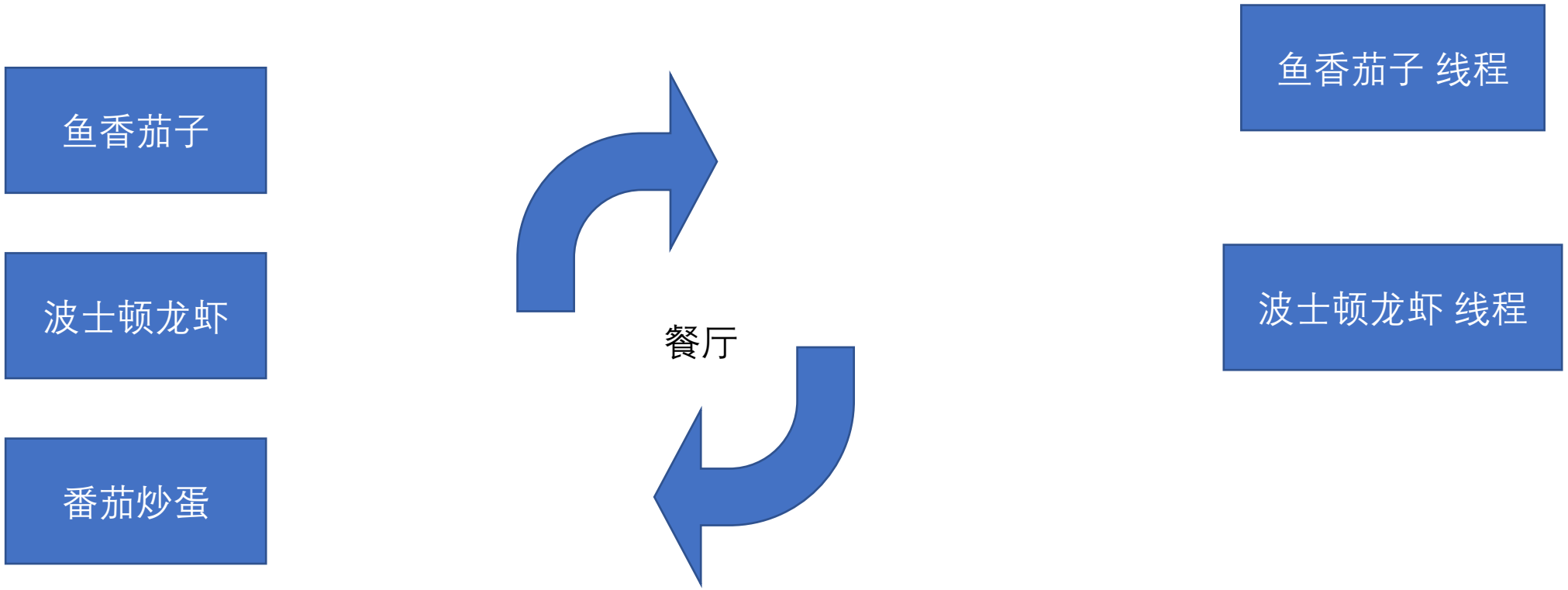
波士顿龙虾

鱼香茄子 线程

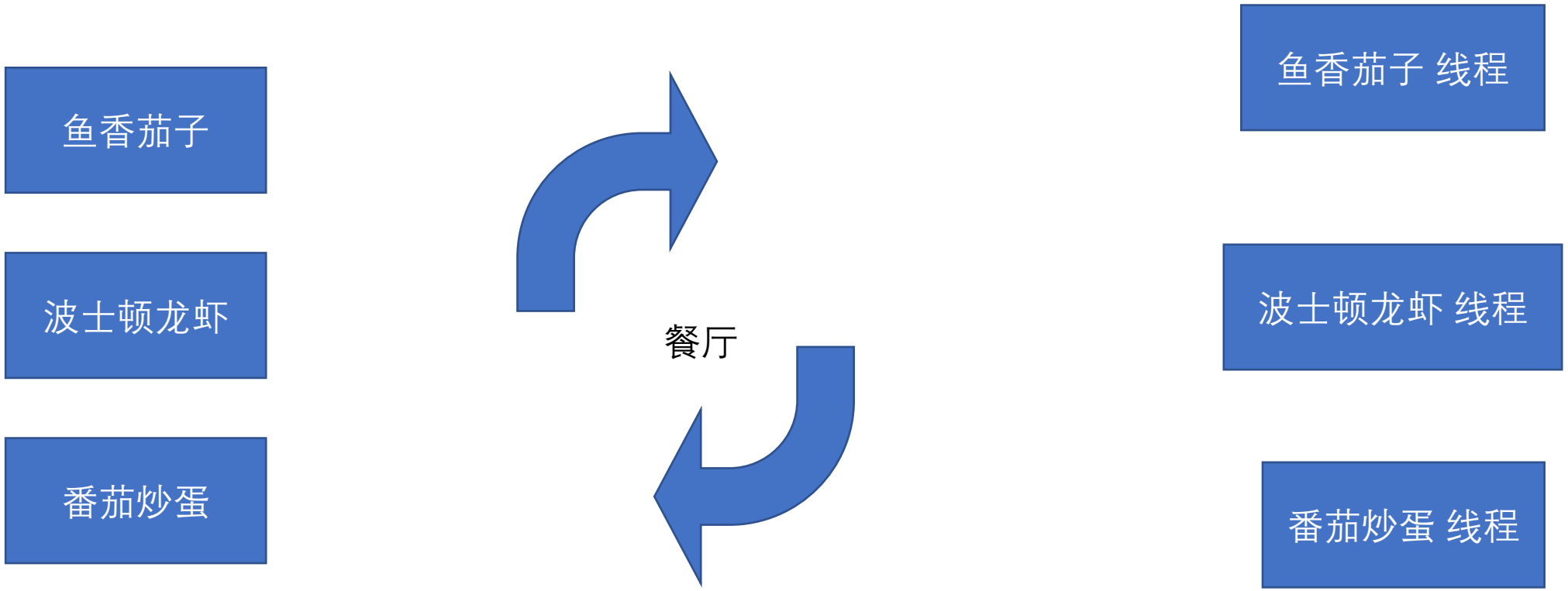
事件循环



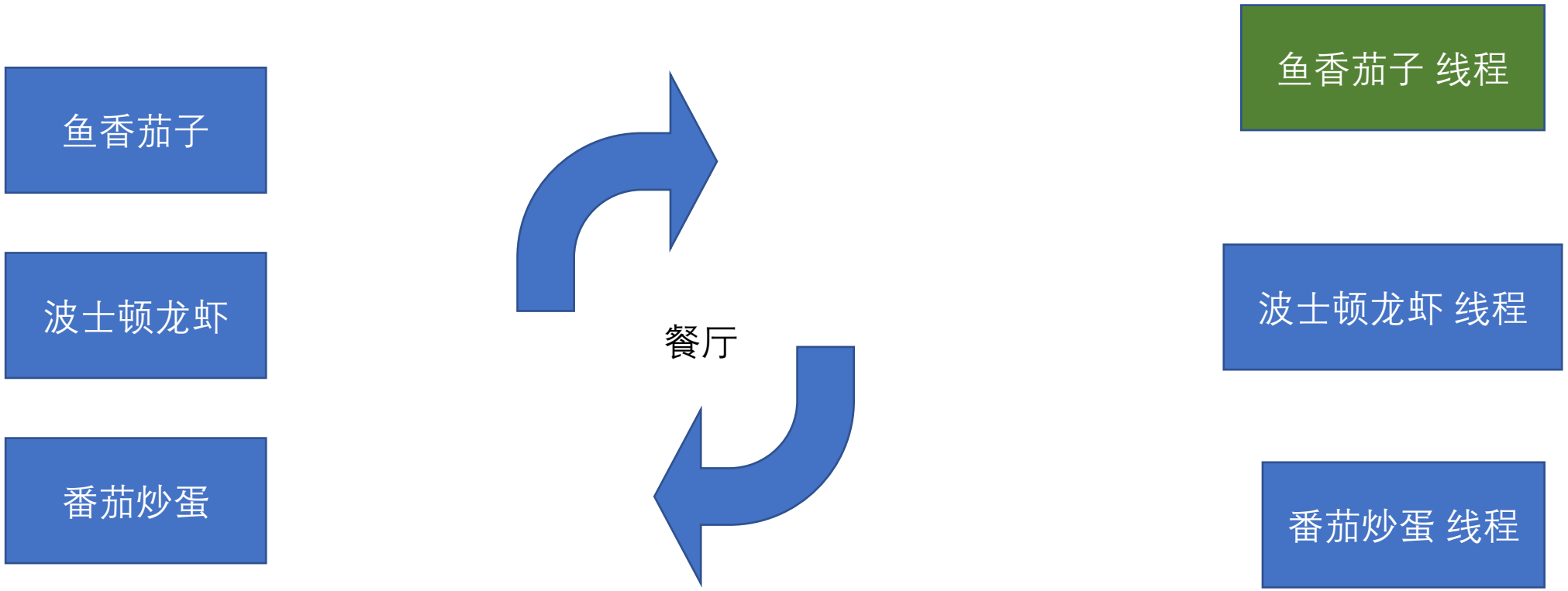
事件循环



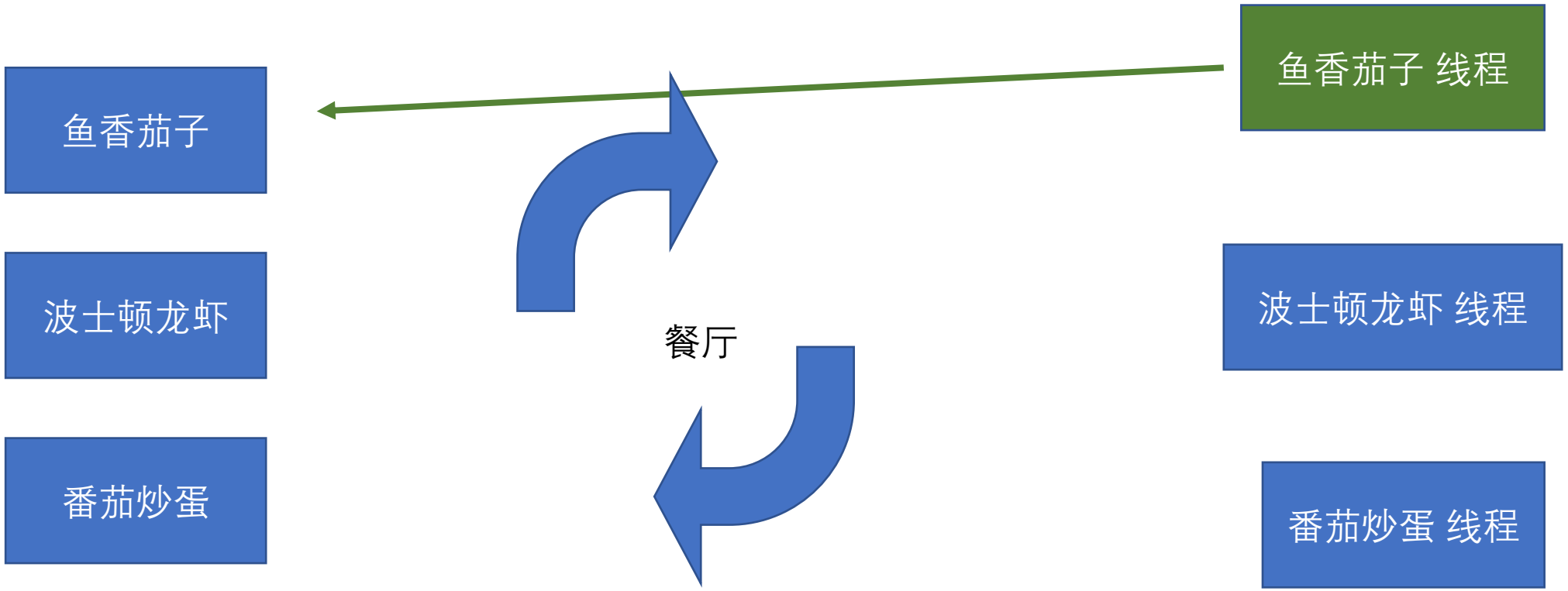
事件循环



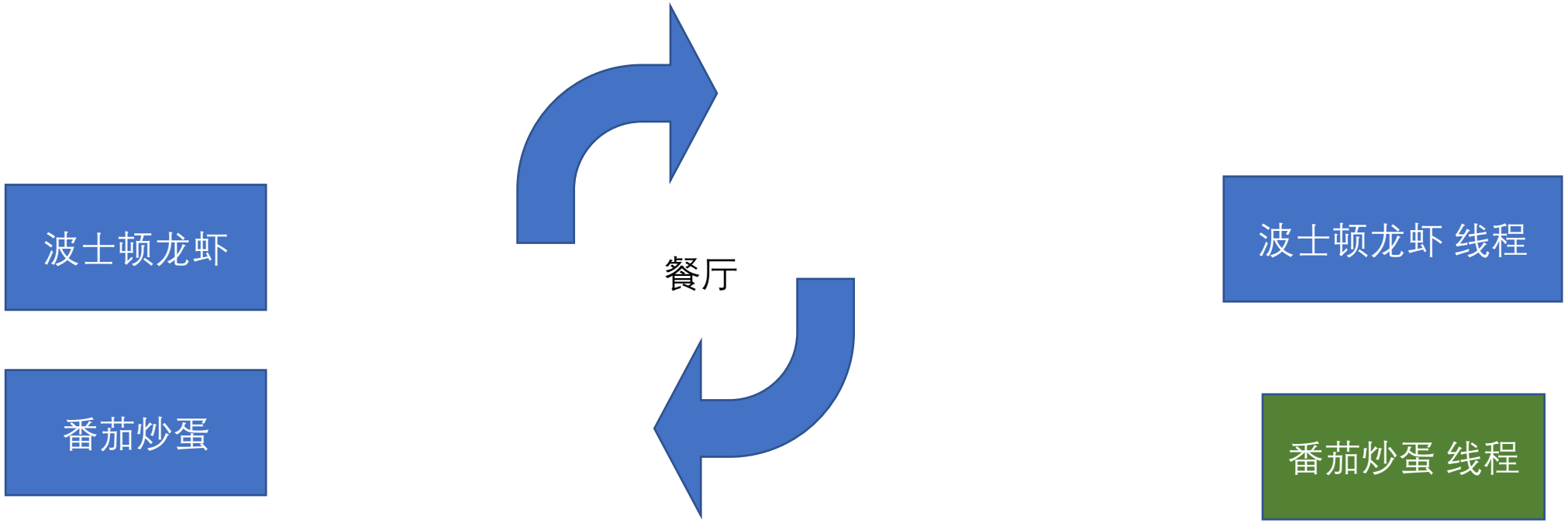
事件循环



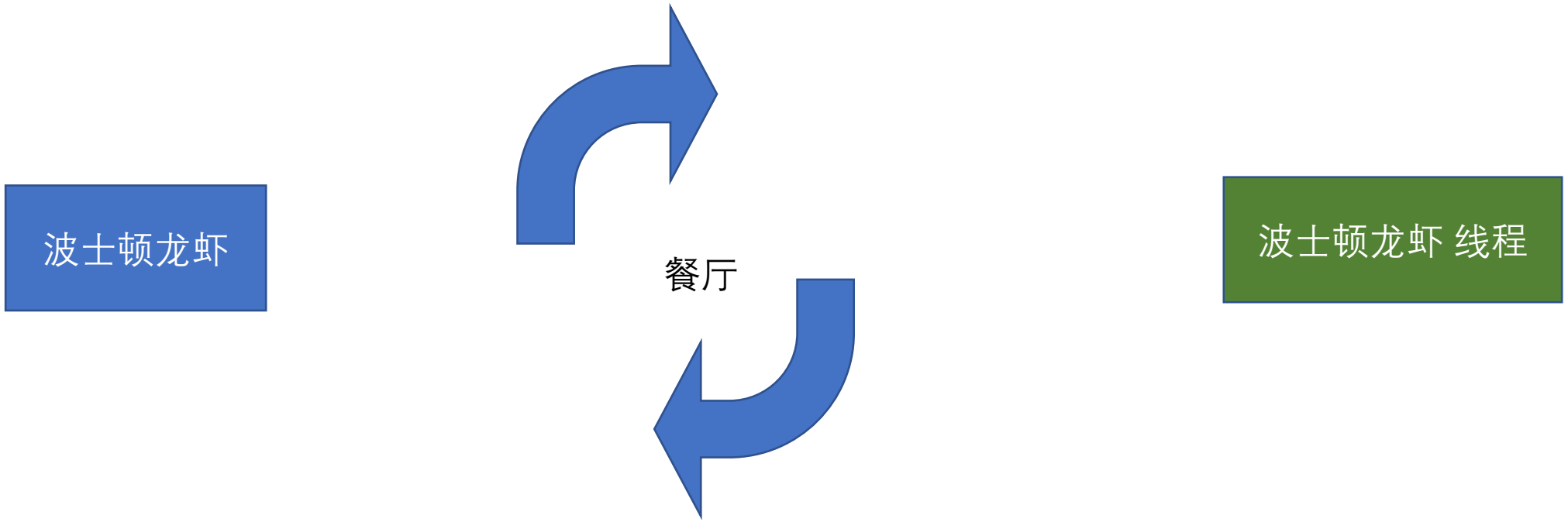
事件循环



事件循环



事件循环

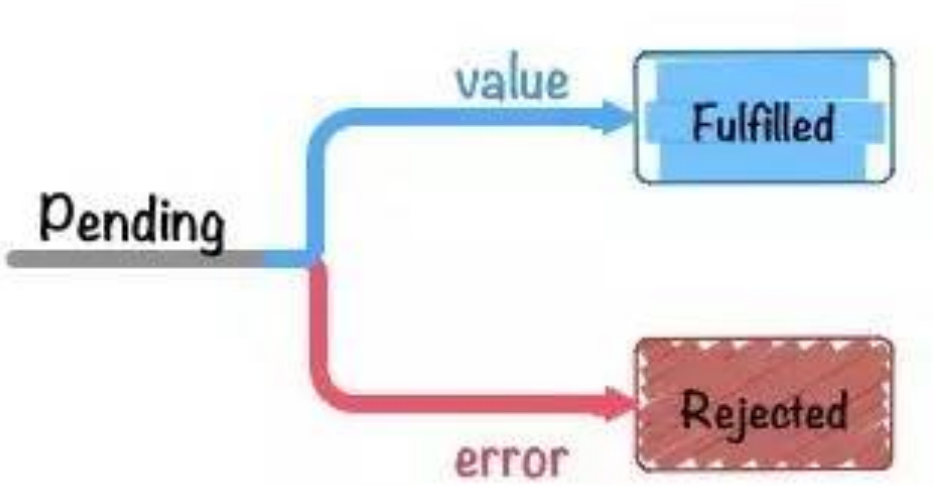


事件循环

Node.js 异步编程 - Promise

Node.js 异步编程 - Promise

- Promise
  - 当前事件循环得不到的结果，但未来的事件循环会给到你结果
  - 是一个状态机
    - pending
    - fulfilled / resolved
    - rejected



Node.js 异步编程 - Promise

- .then 和 .catch
  - resolved 状态的 Promise 会回调后面的第一个 .then
  - rejected 状态的 Promise 会回调后面的第一个 .catch
  - 任何一个 rejected 状态且后面没有 .catch 的 Promise，都会造成 浏览器 /node 环境的全局错误

Node.js 异步编程 - Promise

- 执行 then 和 catch 会返回一个新 Promise，该 Promise 最终状态根据 then 和 catch 的回调函数的执行结果决定
  - 如果回调函数最终是 throw，该 Promise 是 rejected 状态
  - 如果回调函数最终是 return，该 Promise 是 resolved 状态

Node.js 异步编程 - Promise

- 执行 then 和 catch 会返回一个新 Promise，该 Promise 最终状态根据 then 和 catch 的回调函数的执行结果决定
  - 如果回调函数最终是 throw，该 Promise 是 rejected 状态
  - 如果回调函数最终是 return，该 Promise 是 resolved 状态
  - 但如果回调函数最终 return 了一个 Promise，该 Promise 会和回调函数 return 的 Promise 状态保持一致

Node.js 异步编程 - async/await

Node.js 异步编程 - async/await

- async/await
  - async function 是 Promise 的语法糖封装

Node.js 异步编程 - async/await

- async/await
  - async function 是 Promise 的语法糖封装
  - 异步编程的终极方案 - 以同步的方式写异步

Node.js 异步编程 - async/await

- async/await
  - async function 是 Promise 的语法糖封装
  - 异步编程的终极方案 - 以同步的方式写异步
    - await 关键字可以“暂停”async function的执行
    - await 关键字可以以同步的写法获取 Promise 的执行结果
    - try-catch 可以获取 await 所得到的错误

Node.js 异步编程 - async/await

- async/await
  - async function 是 Promise 的语法糖封装
  - 异步编程的终极方案 - 以同步的方式写异步
    - await 关键字可以“暂停”async function的执行
    - await 关键字可以以同步的写法获取 Promise 的执行结果
    - try-catch 可以获取 await 所得到的错误
  - 一个穿越事件循环存在的 function

什么是 HTTP 服务？

什么是 HTTP 服务？

- HTTP 是什么？

什么是 HTTP 服务？

- HTTP 是什么？
  - 应用层协议
  - 五层网络协议

什么是 HTTP 服务？

- HTTP 是什么？
  - 极客主编的一份神秘大礼
  - 经过打包之后
  - 送到了快递员手里
  - 在茫茫世界找到了目的地
  - 然后搭着快递车
  - 通过高速公路送到了目的地

什么是 HTTP 服务？

- HTTP 是什么？
  - 极客主编的一份神秘大礼
  - 经过打包之后
  - 送到了快递员手里
  - 在茫茫世界找到了目的地
  - 然后搭着快递车
  - 通过高速公路送到了目的地

  - 变成了手上的一个神秘大礼
  - 极客粉丝把包裹拆开之后
  - 快递员送到了家里
  - 看到了要送到哪里
  - 上面卸下来了一个包裹
  - 一辆快递车开进了物流中心

什么是 HTTP 服务？

- HTTP 是什么？
  - 极客主编的一份神秘大礼
  - 经过打包之后
  - 送到了快递员手里
  - 在茫茫世界找到了目的地
  - 然后搭着快递车
  - 通过高速公路送到了目的地

  - 变成了手上的一个神秘大礼
  - 极客粉丝把包裹拆开之后
  - 快递员送到了家里
  - 看到了要送到哪里
  - 上面卸下来了一个包裹
  - 一辆快递车开进了物流中心

什么是 HTTP 服务？

- 一个网页请求，包含两次 HTTP 包交换：
  - 浏览器向 HTTP 服务器发送请求 HTTP 包
  - HTTP 服务器向浏览器返回 HTTP 包

- 以极客时间首页为例，分析 HTTP 格式。

什么是 HTTP 服务？

- HTTP 服务要做什么事情?
  - 解析进来的 HTTP 请求报文
  - 返回对应的 HTTP 返回报文

实现一个 HTTP 服务

实现网页版石头剪刀布

HTTP 服务框架：Express

Express

- 要了解一个框架，最好的方法是
  - 了解它的关键功能
  - 推导出它要解决的问题是什么

Express

- 核心功能：
  - 路由

Express

- 核心功能：
  - 路由
  - request/response 简化
    - request: pathname、query 等
    - response: send()、json()、jsonp() 等

Express

- 核心功能：
  - 路由
  - request/response 简化
    - request: pathname、query 等
    - response: send()、json()、jsonp() 等
  - 中间件
    - 更好地组织流程代码
    - 异步会打破 Express 的洋葱模型

HTTP 服务框架：Koa

Koa

- 核心功能：
  - 比 Express 更极致的 request/response 简化
    - ctx.status = 200
    - ctx.body = 'hello world'

Koa

- 核心功能：
  - 使用 async function 实现的中间件
    - 有“暂停执行”的能力
    - 在异步的情况下也符合洋葱模型

Koa

- 核心功能：
  - 精简内核，所有额外功能都移到中间件里实现。

Koa

- Express vs Koa
  - Express 门槛更低，Koa 更强大优雅。
  - Express 封装更多东西，开发更快速，Koa 可定制型更高。

Koa

- Express vs Koa
  - Express 门槛更低，Koa 更强大优雅。
  - Express 封装更多东西，开发更快速，Koa 可定制型更高。

  - 孰“优”孰“劣”？

Koa

- Express vs Koa
  - Express 门槛更低，Koa 更强大优雅。
  - Express 封装更多东西，开发更快速，Koa 可定制型更高。

  - 孰“优”孰“劣”？
    - 框架之间其实没有优劣之分
    - 不同的框架有不同的适用场景

RPC 调用

RPC 调用

- Remote Procedure Call（远程过程调用）

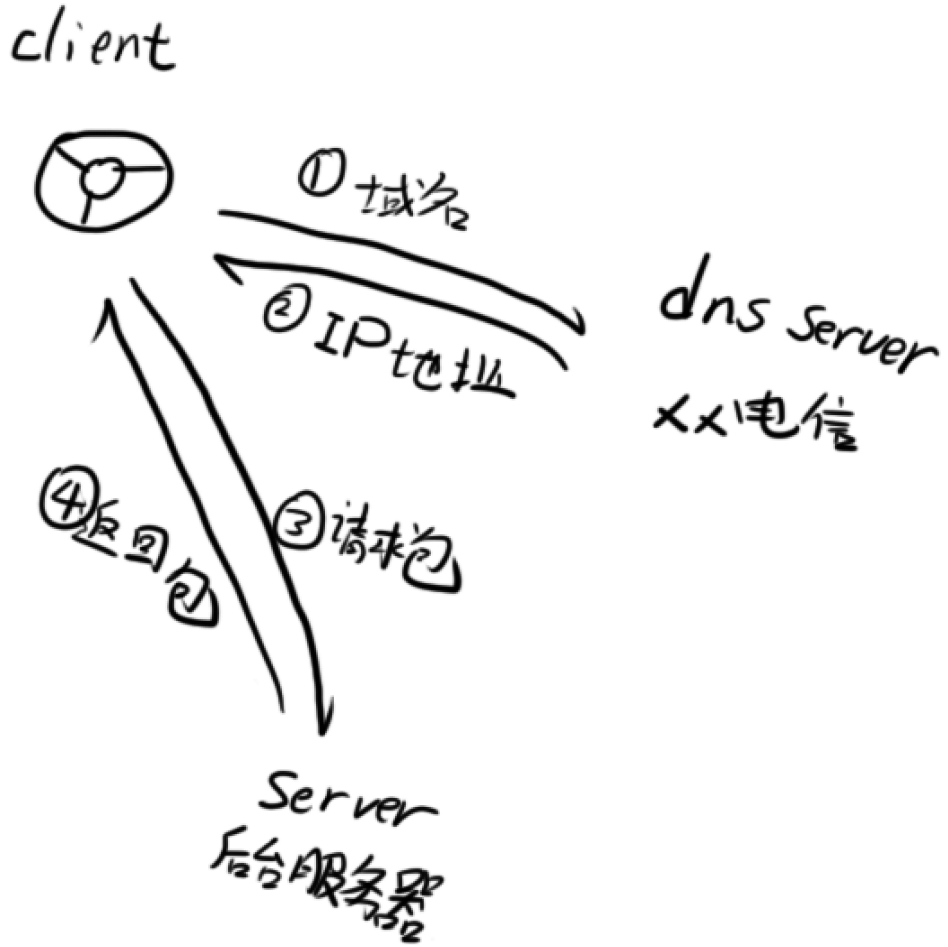
- 和 Ajax 有什么相同点？
  - 都是两个计算机之间的网络通信
  - 需要双方约定一个数据格式

RPC 调用

- 和 Ajax 有什么不同点？
  - 不一定使用 DNS 作为寻址服务
  - 应用层协议一般不使用 HTTP
  - 基于 TCP 或 UDP 协议

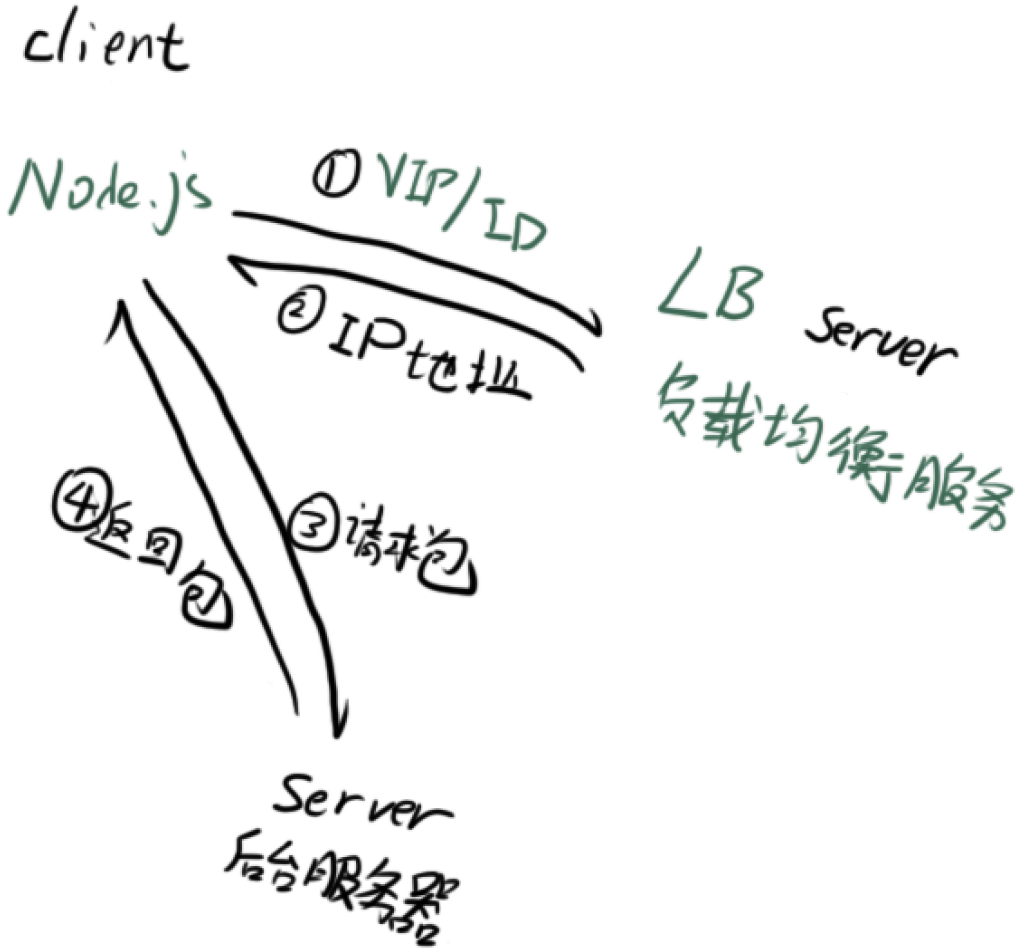
RPC 调用

- 寻址/负载均衡
  - Ajax：使用 DNS 进行寻址



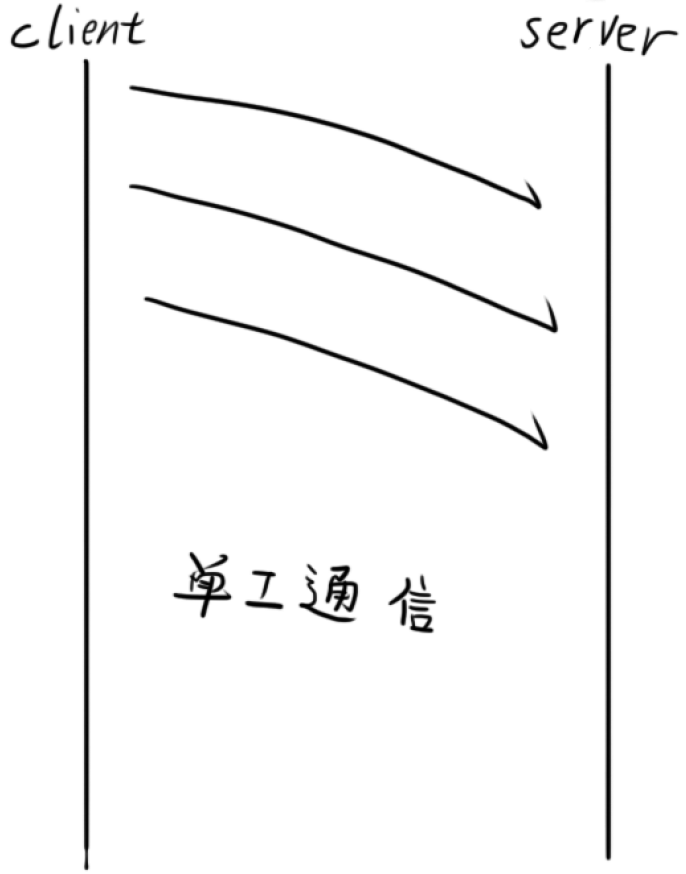
RPC 调用

- 寻址/负载均衡
  - RPC：使用特有服务进行寻址



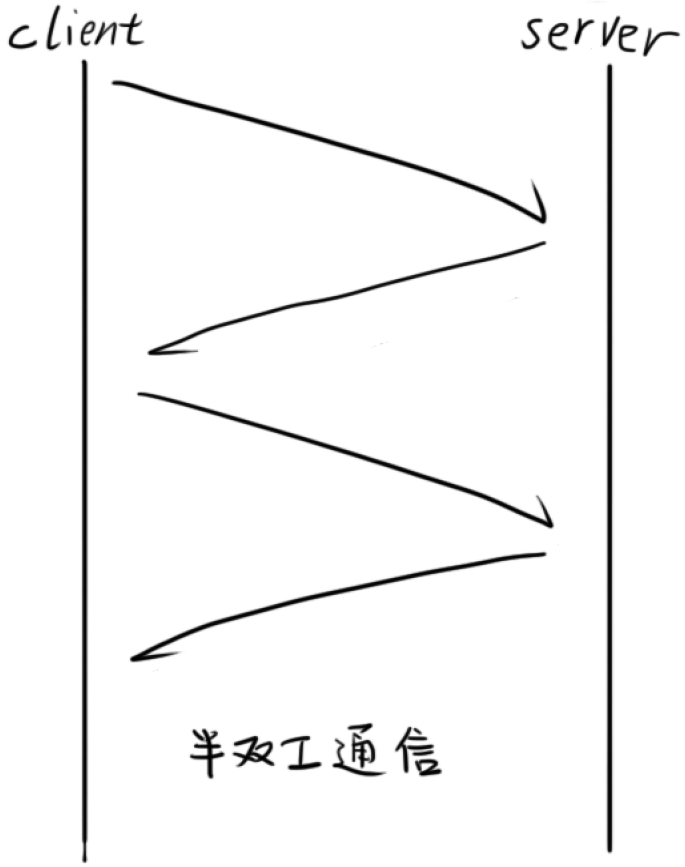
RPC 调用

- TCP 通信方式
  - 单工通信



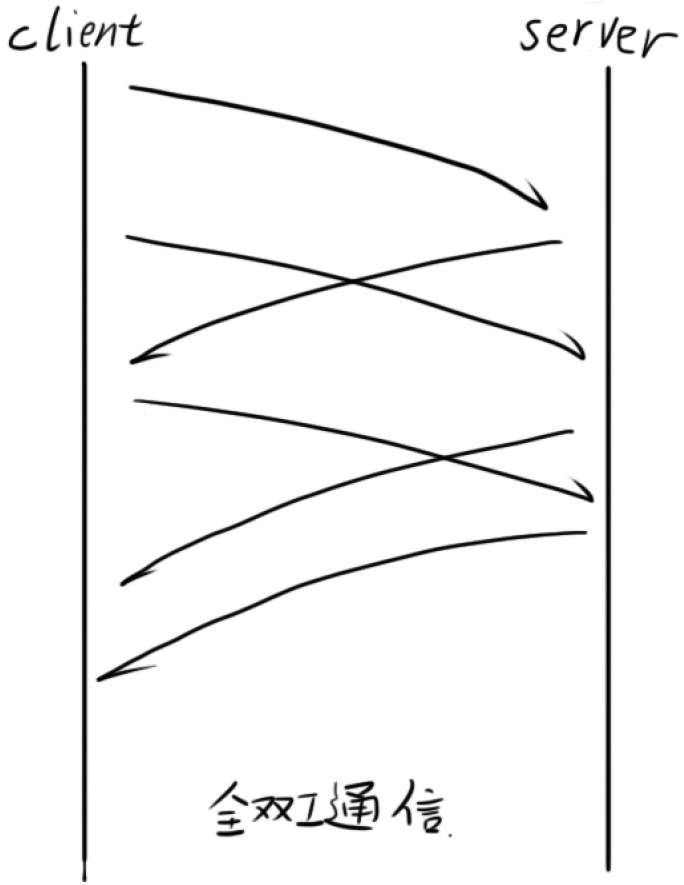
RPC 调用

- TCP 通信方式
  - 单工通信
  - 半双工通信



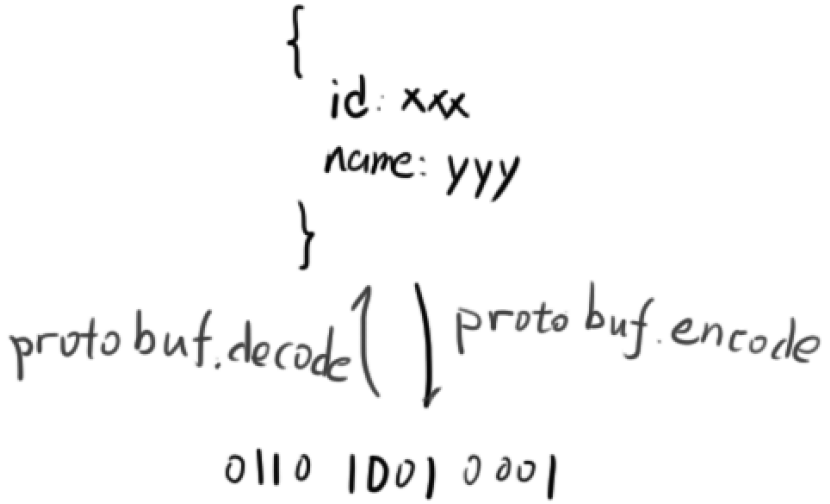
RPC 调用

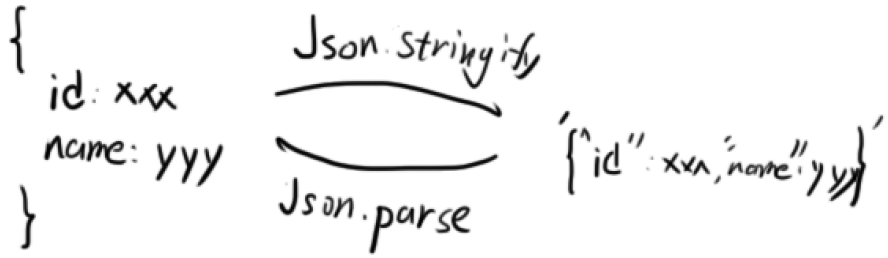
- TCP 通信方式
  - 单工通信
  - 半双工通信
  - 全双工通信



RPC 调用

- 二进制协议
  - 更小的数据包体积
  - 更快的编解码速率





Node.js Buffer 编解码二进制数据包

Node.js Buffer 编解码二进制数据包

- 大小端问题
  - 几个 Byte 里，高位与低位的编排顺序不同。
- 处理方法与 string 接近
  - 使用 concat 而不是 + 来避免 utf-8 字符拼接问题。

Node.js Buffer 编解码二进制数据包

- Protocol Buffer
  - Google 研发的二进制协议编解码库
  - 通过协议文件控制 Buffer 的格式
    - 更直观
    - 更好维护
    - 更便于合作

Node.js net 搭建多路复用的 RPC 通道

Node.js net 模块

- 单工/半双工的通信通道搭建

Node.js net 模块

- 全双工的通信通道搭建
  - 关键在于应用层协议需要有标记包号的字段
  - 处理以下情况，需要有标记包长的字段
    - 粘包
    - 不完整包
  - 错误处理

极客时间

扫码试看/订阅

《Node.js 开发实战》视频课程