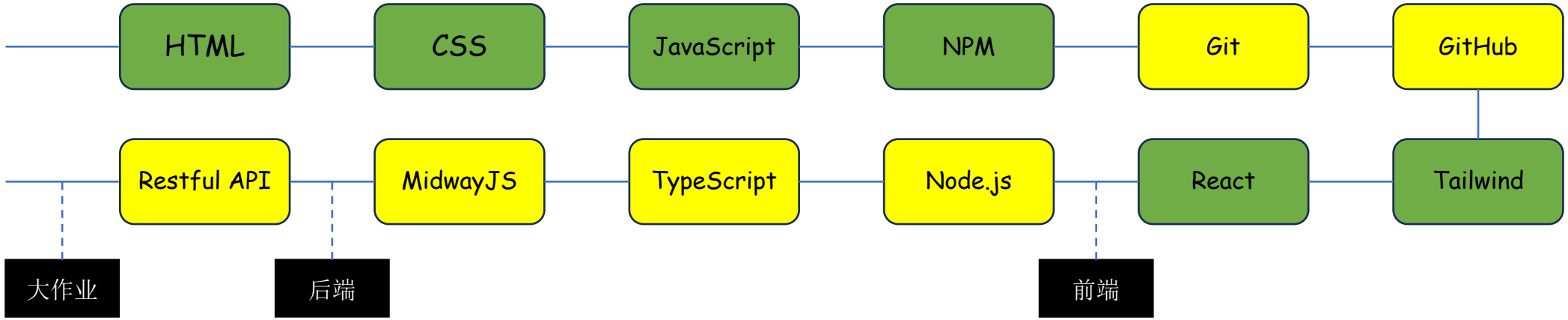
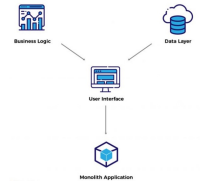
Web 后端开发技术

章许帆

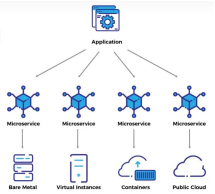

HTML
CSS
JavaScript
NPM
Git
GitHub
Restful API
MidwayJS
TypeScript
Node.js
React
Tailwind
大作业
后端
前端

Web 后端开发技术

单服务架构



微服务架构



后端服务架构

控制器

异步编程模型

依赖注入

应用与上下文

Node.js Midway框架

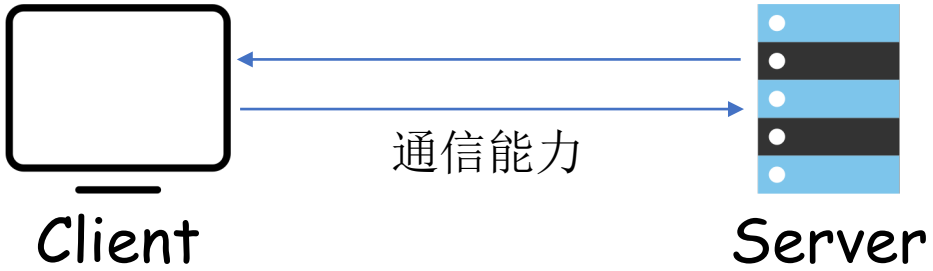
Midway 框架

POSTMAN

实验: 构建一个Web App 后端

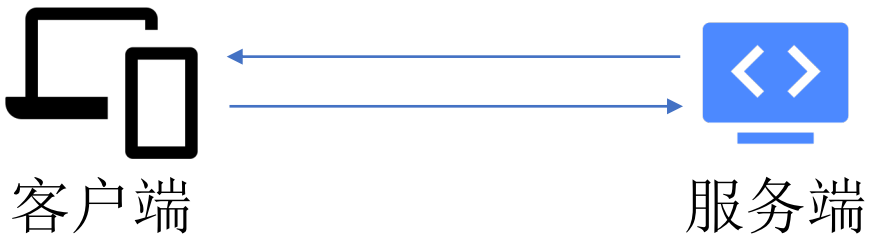
Web 应用架构

Web 2.0 App

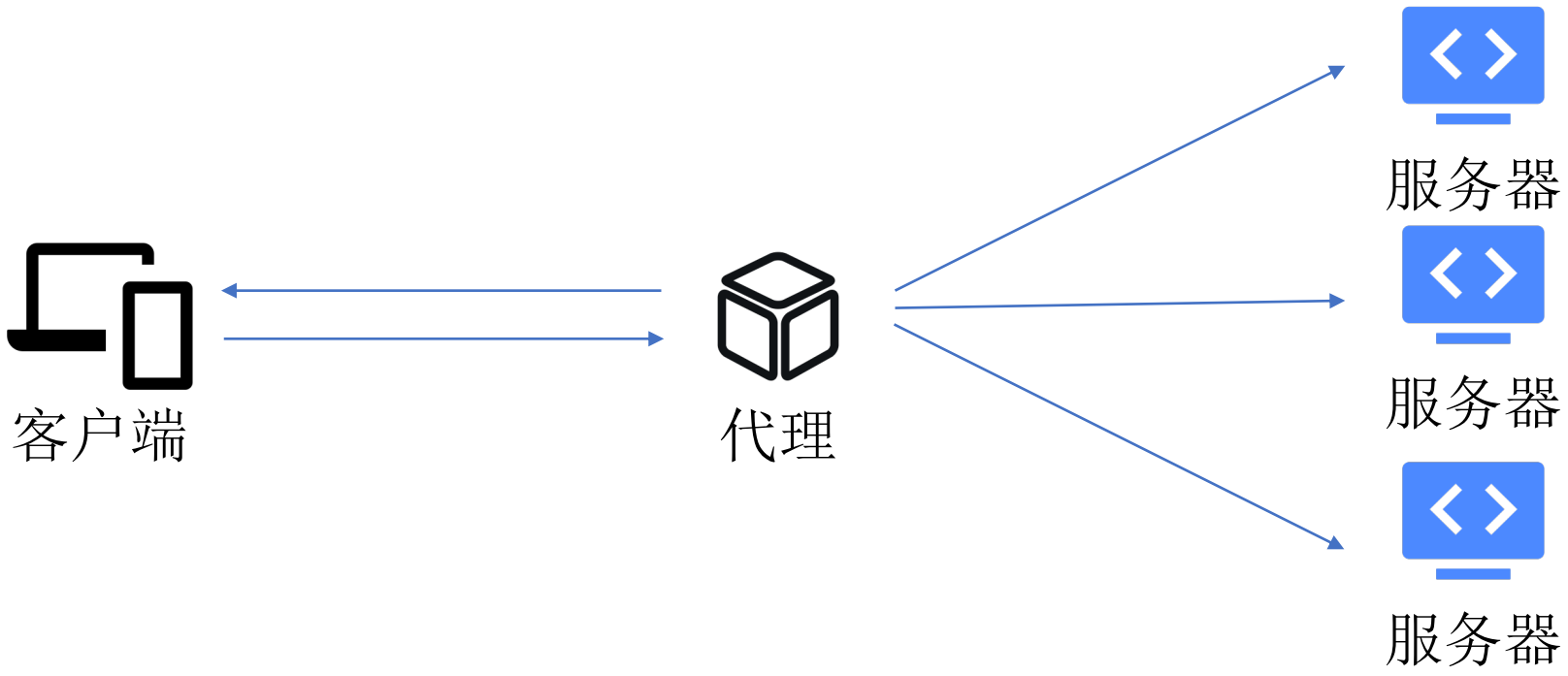


如何实现一个Server端的服务

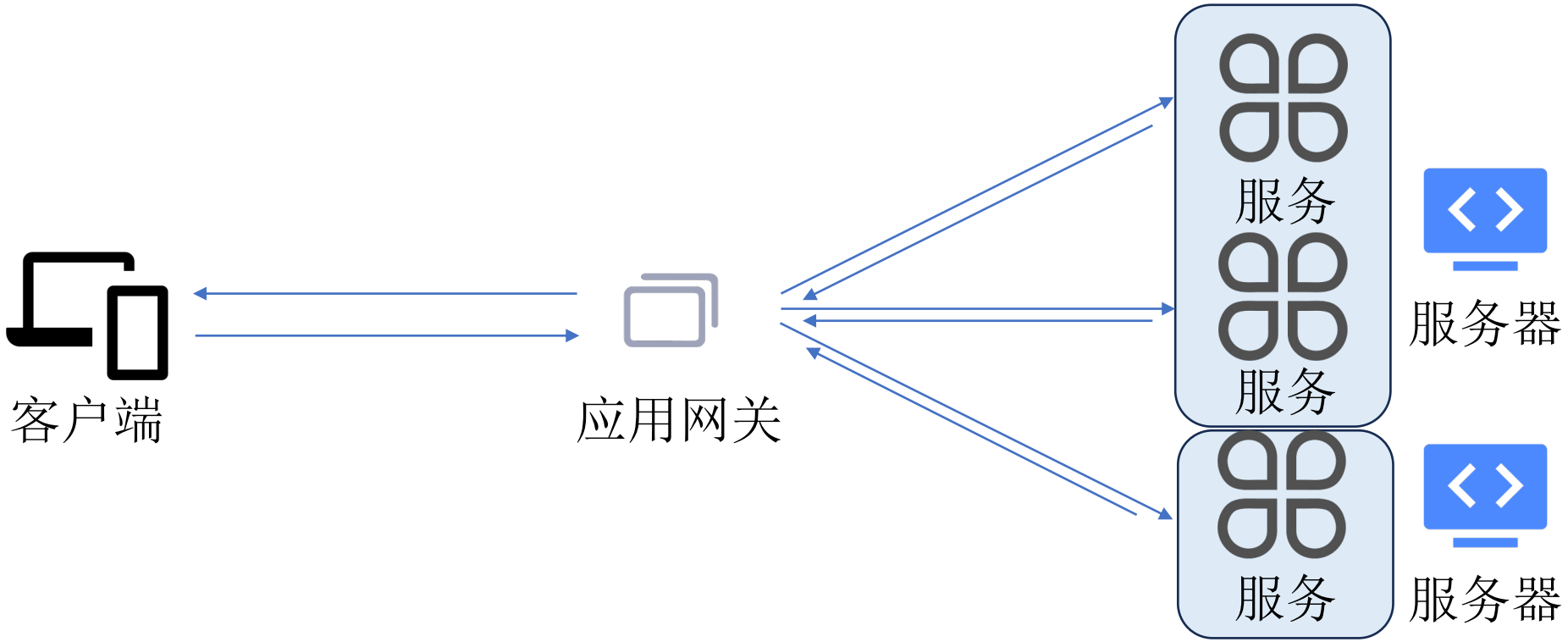
单体架构



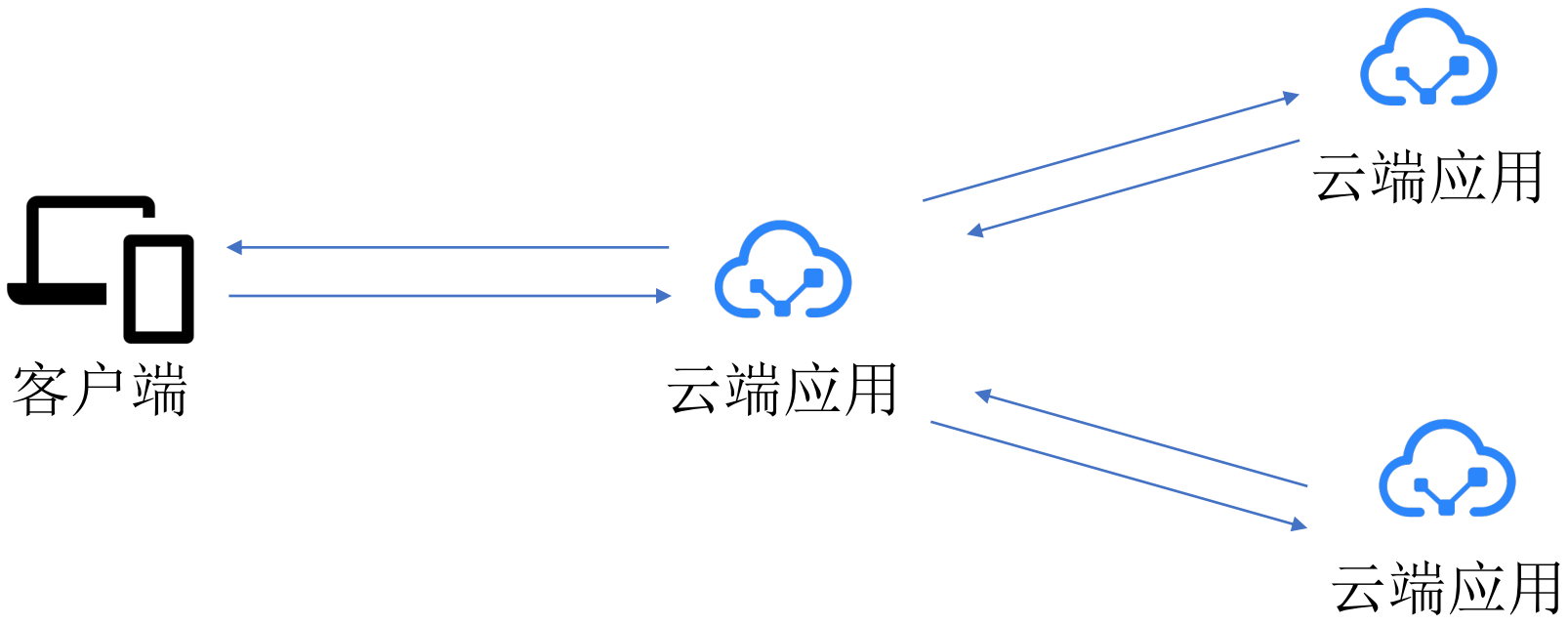
分布式架构



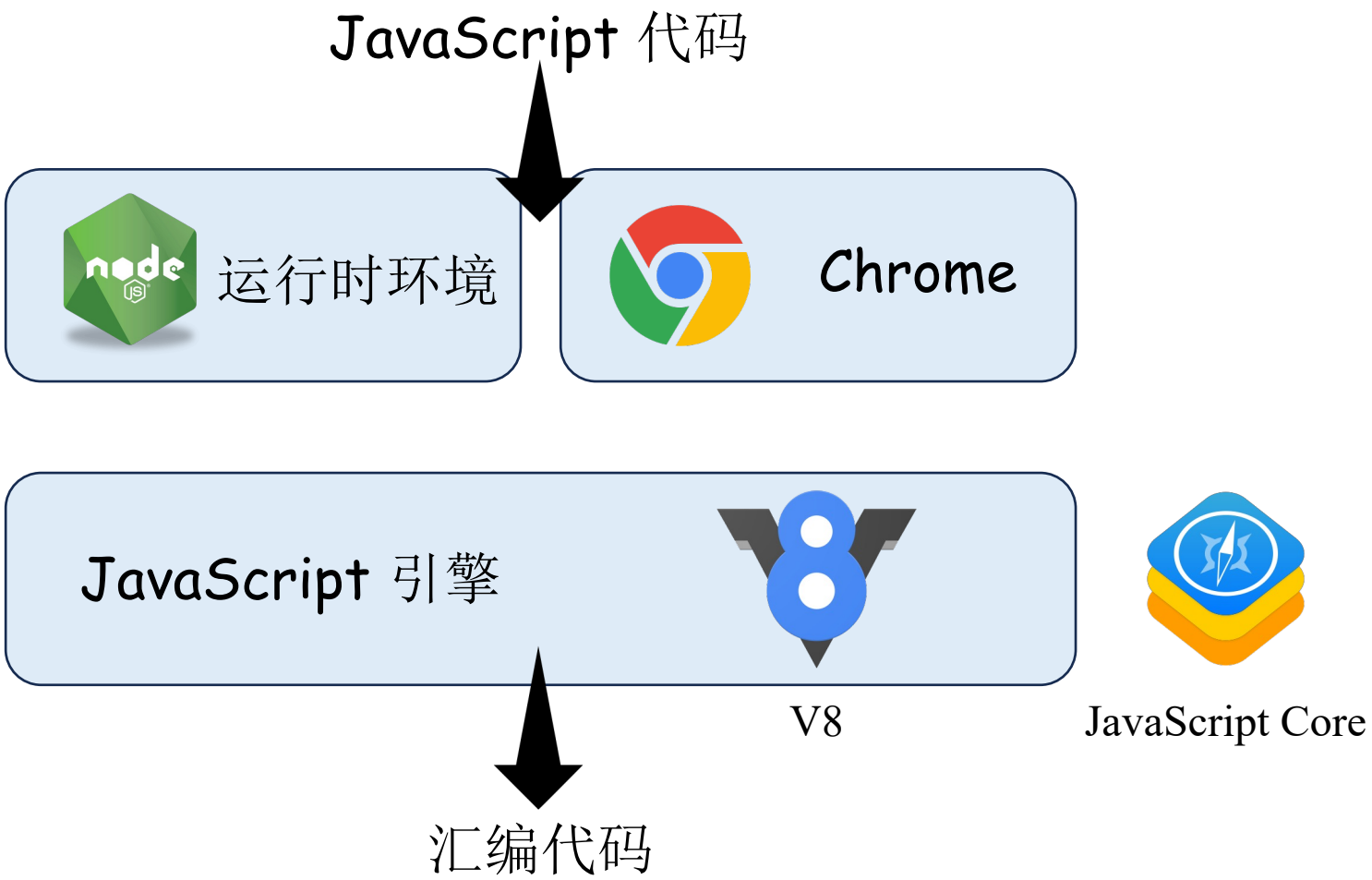
微服务架构



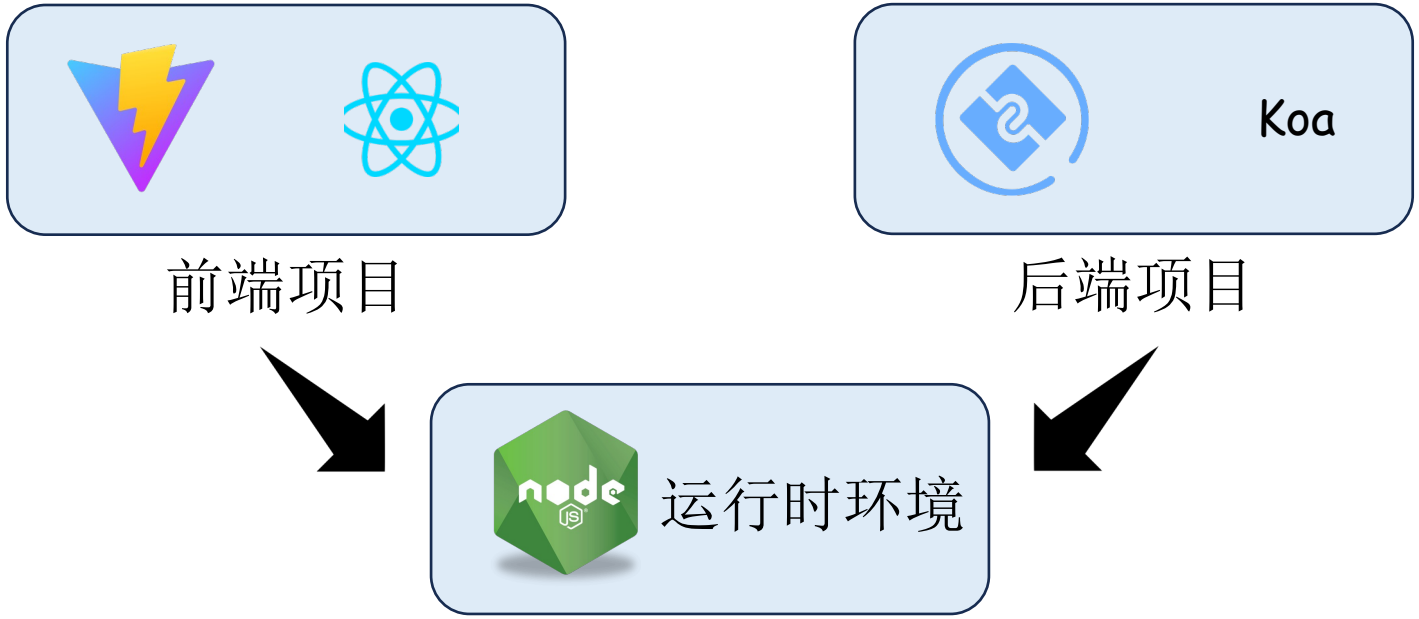
Serverless 架构



Node.js


JavaScript 代码
运行时环境
Chrome
JavaScript 引擎
V8
JavaScript Core
汇编代码

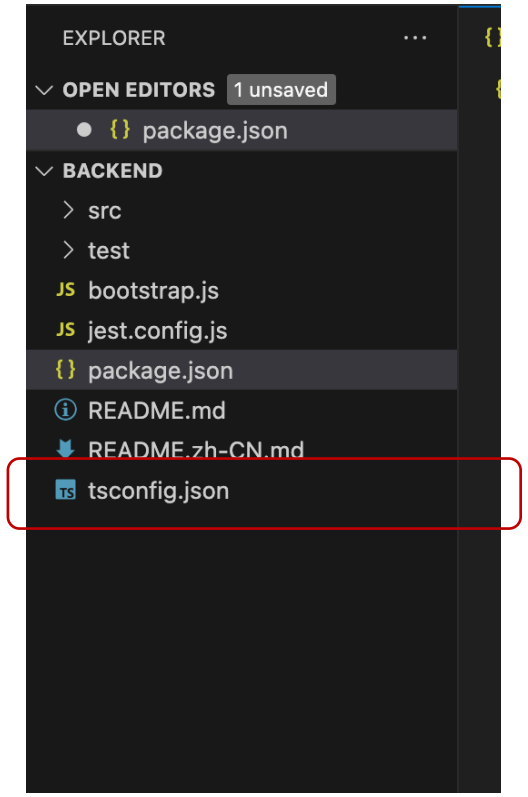
Node.js



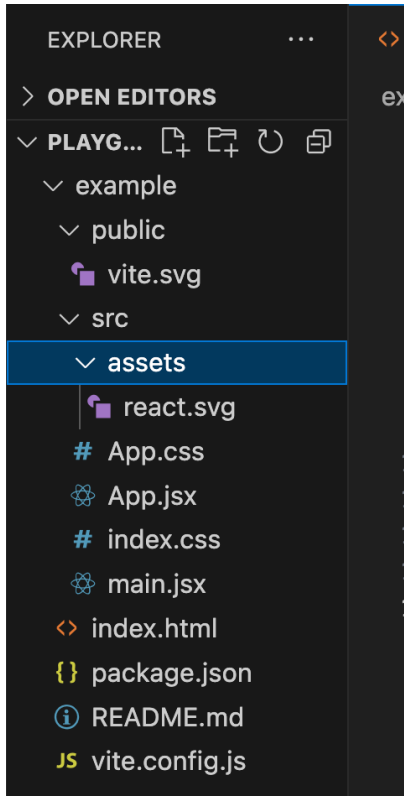
初始化后端代码库

```
[fan@Finleys-MacBook-Pro backend % node -v
 v20.15.0
[fan@Finleys-MacBook-Pro backend % npm init midway
 Need to install the following packages:
 create-midway@1.3.1
[Ok to proceed? (y) y


 > npx
 > create-midway

 ? Hello, traveller.
   Which template do you like? …

  ◎ v3
 ❯ koa-v3 - A web application boilerplate with midway v3(koa)
   egg-v3 - A web application boilerplate with midway v3(egg 2.0)
   faas-v3-new - A serverless application boilerplate with midway v3(faas)
   component-v3 - A midway component boilerplate for v3
   quick-start - A midway quickstart example for v3

  ◎ v3-esm
   koa-v3-esm - A web application boilerplate with midway v3(koa)

  ◎ v2
   web - A web application boilerplate with midway and Egg.js
   koa - A web application boilerplate with midway and koa
```

package.json



后端



前端

TypeScript

TS index.ts 1

src > demo > TS index.ts > ...

```ts
const user = {
    name: "Xufan Zhang",
    email: "xufan.zhang@outlook.com",
}

console.log(user.age);
```

JS index.js

src > demo > JS index.js > ...

```js
const user = {
    name: "Xufan Zhang",
    email: "xufan.zhang@outlook.com",
}

console.log(user.age);
```

有类型的JavaScript

TypeScript数据类型

- number, boolean, string
- number[], Array<number>
- any
- interface, union...

```
1   let a: number;
2
3   let b: string;
4
5   let c: boolean;
6
7   let num_arr: number[];
8
9   let string_array: Array<string>;
10
11  let obj: any;
```

TypeScript函数

参数类型　　返回值类型

```typescript
function f1(param_1: number, param_2: string): string {
    return 'Hello';
}
```

TypeScript 函数定义

```javascript
function f1(param_1, param_2) {
    return 'Hello'
}
```

JavaScript 函数定义

TypeScript Class

```
1   export class Task {
2
3       name: string;
4       description: string;
5       createAt: Date;
6
7       constructor(name: string, description: string, createAt: Date) {
8           this.name = name;
9           this.description = description;
10          this.createAt = createAt;
11      }
12
13      public getName(): string {
14          return this.name;
15      }
16  }
```

成员属性、构造方法、成员方法

async 函数 VS. 同步函数

```
3   @Controller('/')
4   export class HomeController {
5     @Get('/')
6     async home(): Promise<object> {
7       return {
8         message: 'Hello Web Development!',
9       };
10    }
11  }
12
```

异步函数返回的是一个未来某一个时刻执行完成的结果

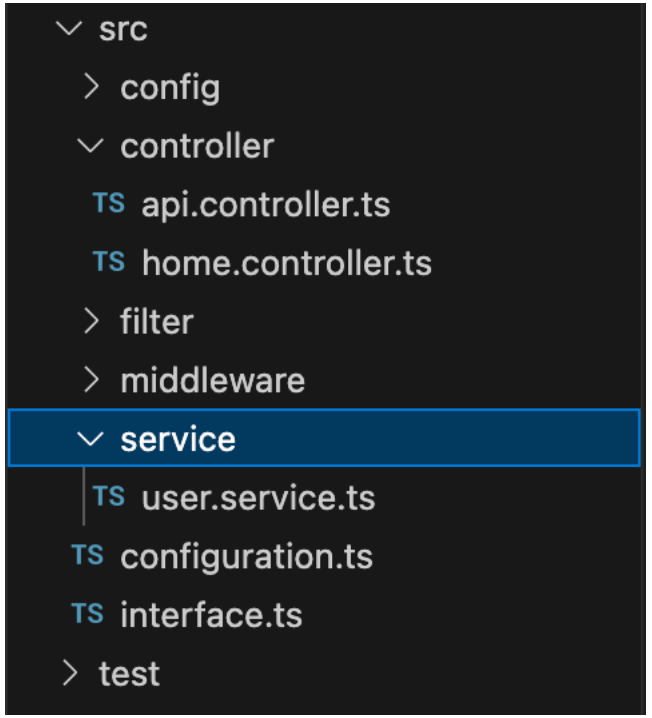
Promise

```
new Promise((resolve, reject) => {})
```

同步与异步函数的差异

```
function add(a: number, b: number): number {
  return a + b;
}

async function minus(a: number, b: number): Promise<number> {
  return a - b;
}

let a = add(1, 2);
let b = minus(3, 4);
let c = await minus(3, 4);

console.log(a, b, c);
```

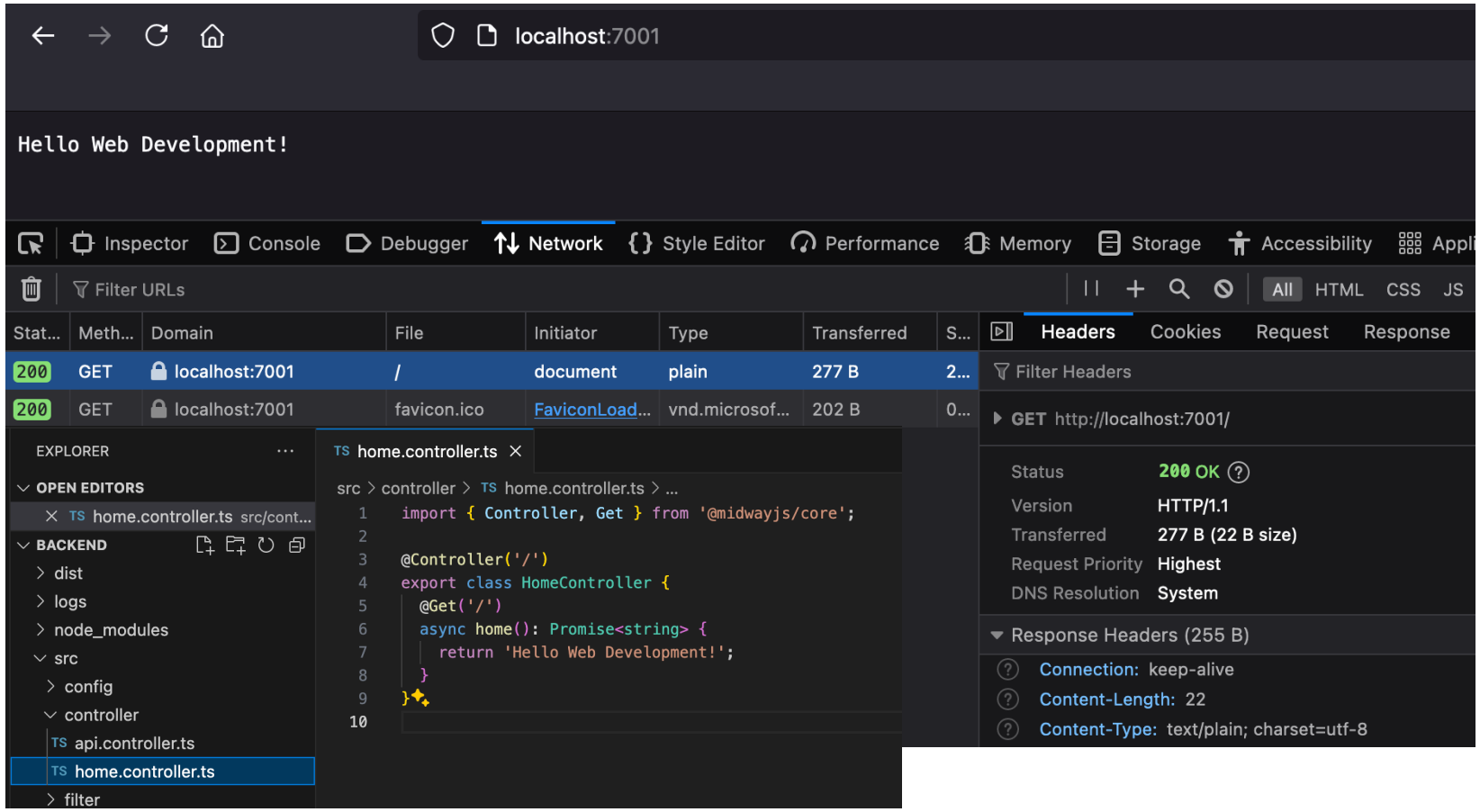
后端代码结构


src
config
controller
TS api.controller.ts
TS home.controller.ts
filter
middleware
service
TS user.service.ts
TS configuration.ts
TS interface.ts
test


Client
HTTP Request
HTTP Response
控制器
(Controller)
服务
(Service)

Controller

负责接收和响应外部请求



text/plain

Controller

负责接收和响应外部请求

TS home.controller.ts

src > controller > TS home.controller.ts > HomeCor

```
1   import { Controller, Get } from '@midwa
2
3   @Controller('/')
4   export class HomeController {
5     @Get('/')
6     async home(): Promise<object> {
7       const result = {
8         message: 'hello web development!'
9       }
10      return result;
11    }
12  }
```

localhost:7001

Reseach keylights 果麦 小范 小刘 小章 瞎捣鼓 Database Java

Pretty-print

{"message":"hello web development!"}

Elements Console Sources Network Performance Memory Application Security Lighth

Preserve log Disable cache No throttling

Filter Invert Hide data URLs Hide extension URLs All Fetch/XHR Doc CSS

3rd-party requests

10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110

Name

localhost

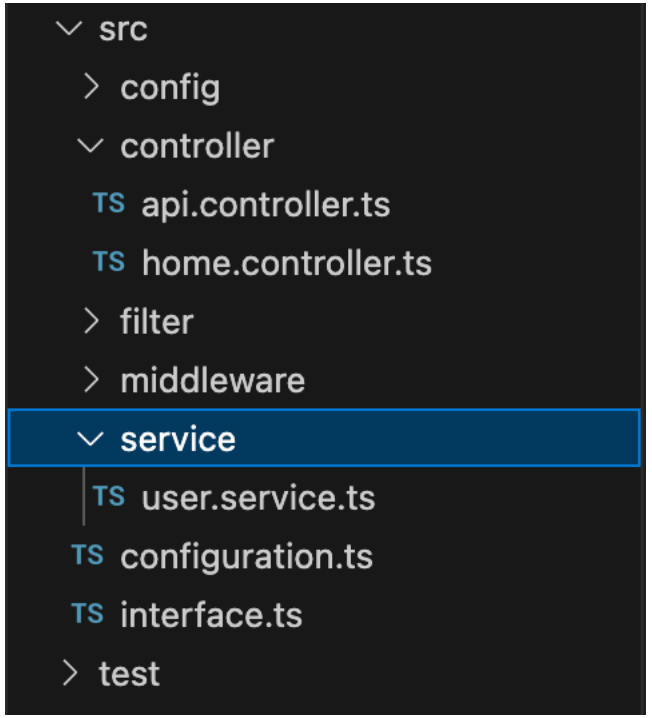
favicon.ico

Headers Preview Response Initiator Timing Cookies

Connection:	keep-alive
Content-Length:	36
Content-Type:	application/json; charset=utf-8

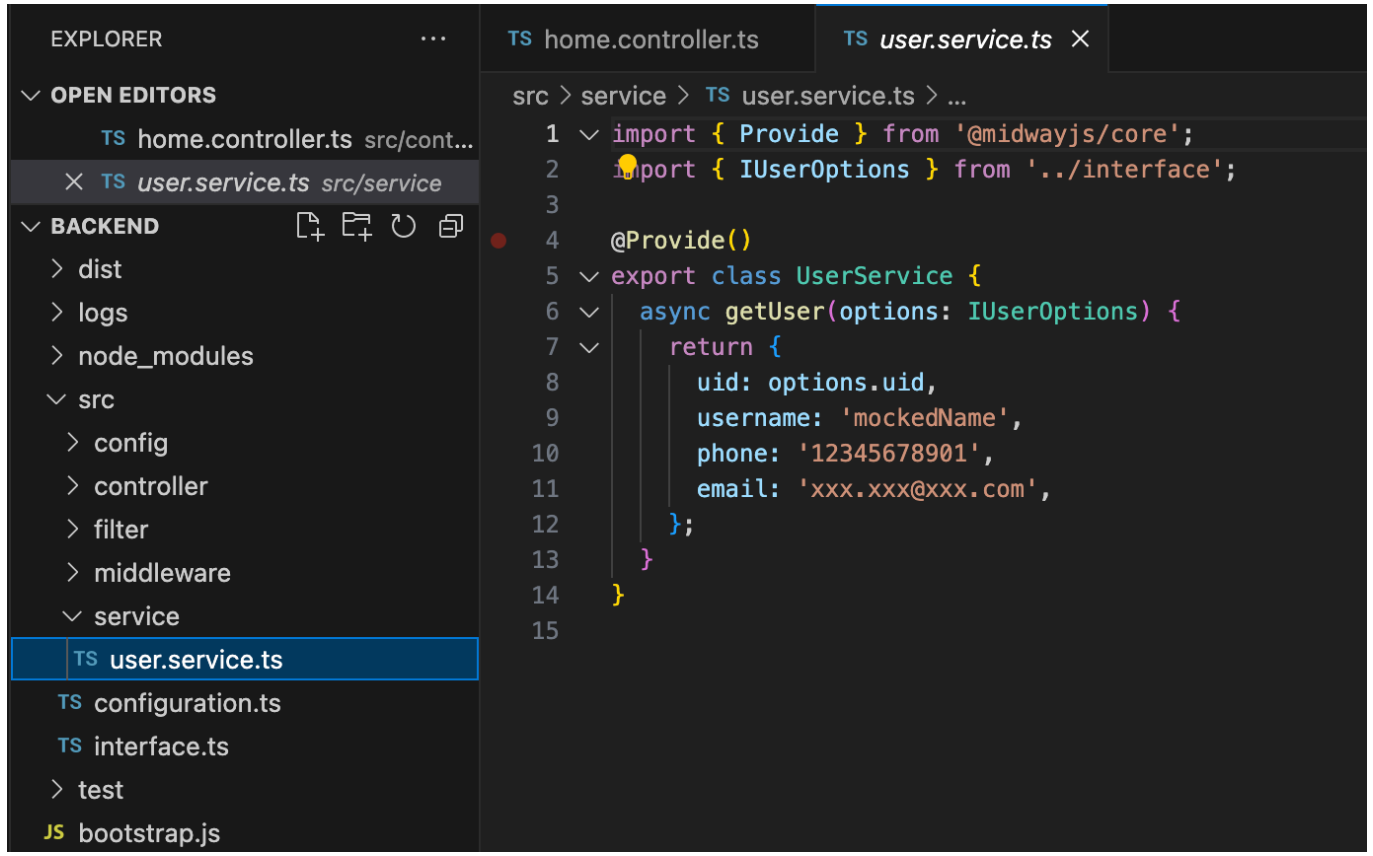
application/json

后端代码结构

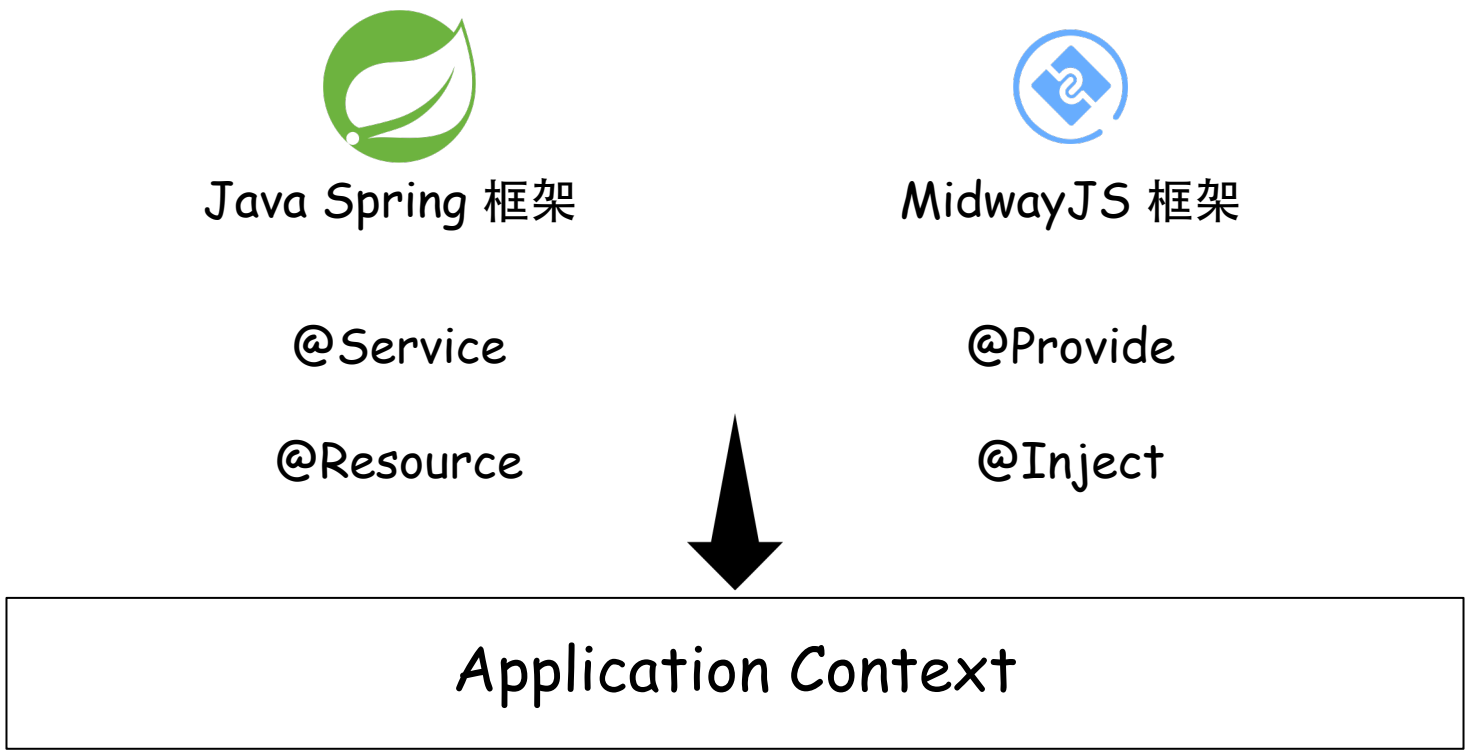

src
config
controller
TS api.controller.ts
TS home.controller.ts
filter
middleware
service
TS user.service.ts
TS configuration.ts
TS interface.ts
test


Client
HTTP Request
HTTP Response
控制器
(Controller)
服务
(Service)

Service


EXPLORER
OPEN EDITORS
home.controller.ts src/cont...
user.service.ts src/service
BACKEND
dist
logs
node_modules
src
config
controller
filter
middleware
service
user.service.ts
configuration.ts
interface.ts
test
bootstrap.js
home.controller.ts
user.service.ts
src > service > user.service.ts > ...
import { Provide } from '@midwayjs/core';
import { IUserOptions } from '../interface';
@Provide()
export class UserService {
async getUser(options: IUserOptions) {
return {
uid: options.uid,
username: 'mockedName',
phone: '12345678901',
email: 'xxx.xxx@xxx.com',
};
}
}

IoC & DI


Java Spring 框架
MidwayJS 框架
@Service
@Provide
@Resource
@Inject
Application Context

Router

http://domain:port/path → function name(param)

```
import { Body, Controller, Post } from "@midwayjs/core";

@Controller('/task')
export class TaskController {

    @Post('/create')
    public async create(@Body() form: {
        name: string;
        description: string;
    }){

    }
}
```

http://domain:port/task/create

文件上传

```
$ npm i @midwayjs/upload@3 --save
```

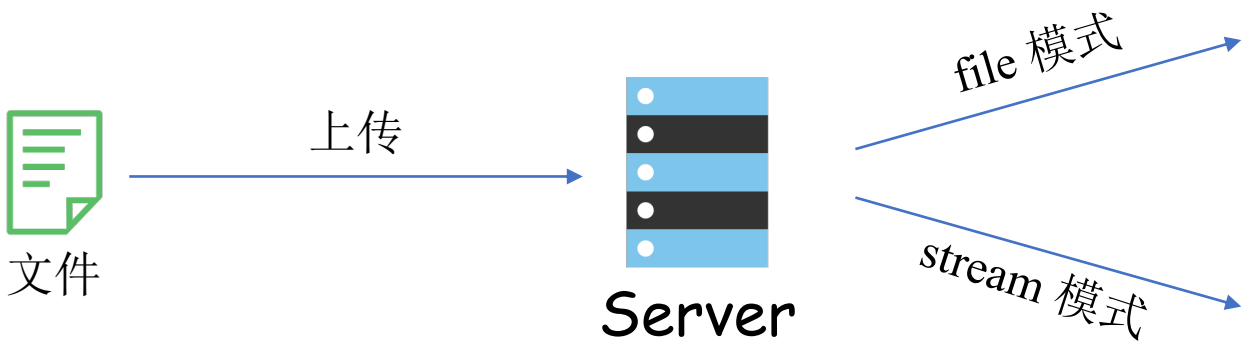
```
import { Configuration } from '@midwayjs/core';
import * as upload from '@midwayjs/upload';

@Configuration({
  imports: [
    // ...other components
    upload
  ],
  // ...
})
export class MainConfiguration {}
```

configuration.ts

```
export default {
  // ...
  upload: {
    // mode: UploadMode, 默认为file，即上传到服务器临时目录，可以配置为 stream
    mode: 'file' , // 'stream'
    // fileSize: string, 最大上传文件大小，默认为 10mb
    fileSize: '10mb',
    // whitelist: string[]，文件扩展名白名单
    whitelist: uploadWhiteList.filter(ext => ext !== '.pdf'),
    // tmpdir: string，上传的文件临时存储路径
    tmpdir: join(tmpdir(), 'midway-upload-files'),
    // cleanTimeout: number，上传的文件在临时目录中多久之后自动删除，默认为 5 分钟
    cleanTimeout: 5 * 60 * 1000,
    // base64: boolean，设置原始body是否是base64格式，默认为false，一般用于腾讯云的兼容
    base64: false,
    // 仅在匹配路径到 /api/upload 的时候去解析 body 中的文件信息
    match: /\/api\/upload/,
  },
}
```

config.xx.ts

file VS. stream




```
{
filename: 'test.pdf', // 文件原名
data: '/var/tmp/xxx.pdf', // mode 为 file 时为服务器临时文件地址
fieldname: 'test1', // 表单 field 名
mimeType: 'application/pdf', // mime
}
```

```
{
filename: 'test.pdf', // 文件原名
data: ReadStream, // mode 为 stream 时为文件的二进制流
fieldname: 'test2', // 表单 field 名
mimeType: 'application/pdf', // mime
}
```

filesystem 'fs'

```
1   import * as fs from 'fs';
2   
3   fs.readFile('data.json', 'utf8', (err, data) => {
```

```
readFile(path: number | fs.PathLike, options: {
encoding?: null; flag?: string; }, callback: (err:
NodeJS.ErrnoException, data: Buffer) => void): void
```

A path to a file. If a URL is provided, it must use the `file:` protocol. If a file descriptor is provided, the underlying file will *not* be closed automatically.

1/4

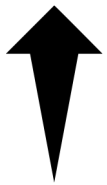
Asynchronously reads the entire contents of a file.

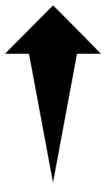
```
fs.readFile('/Users/fan/Documents/Repo/backend/package.json', (error, data) => {

})
```

Callback Function

setCount 以回调函数作为参数

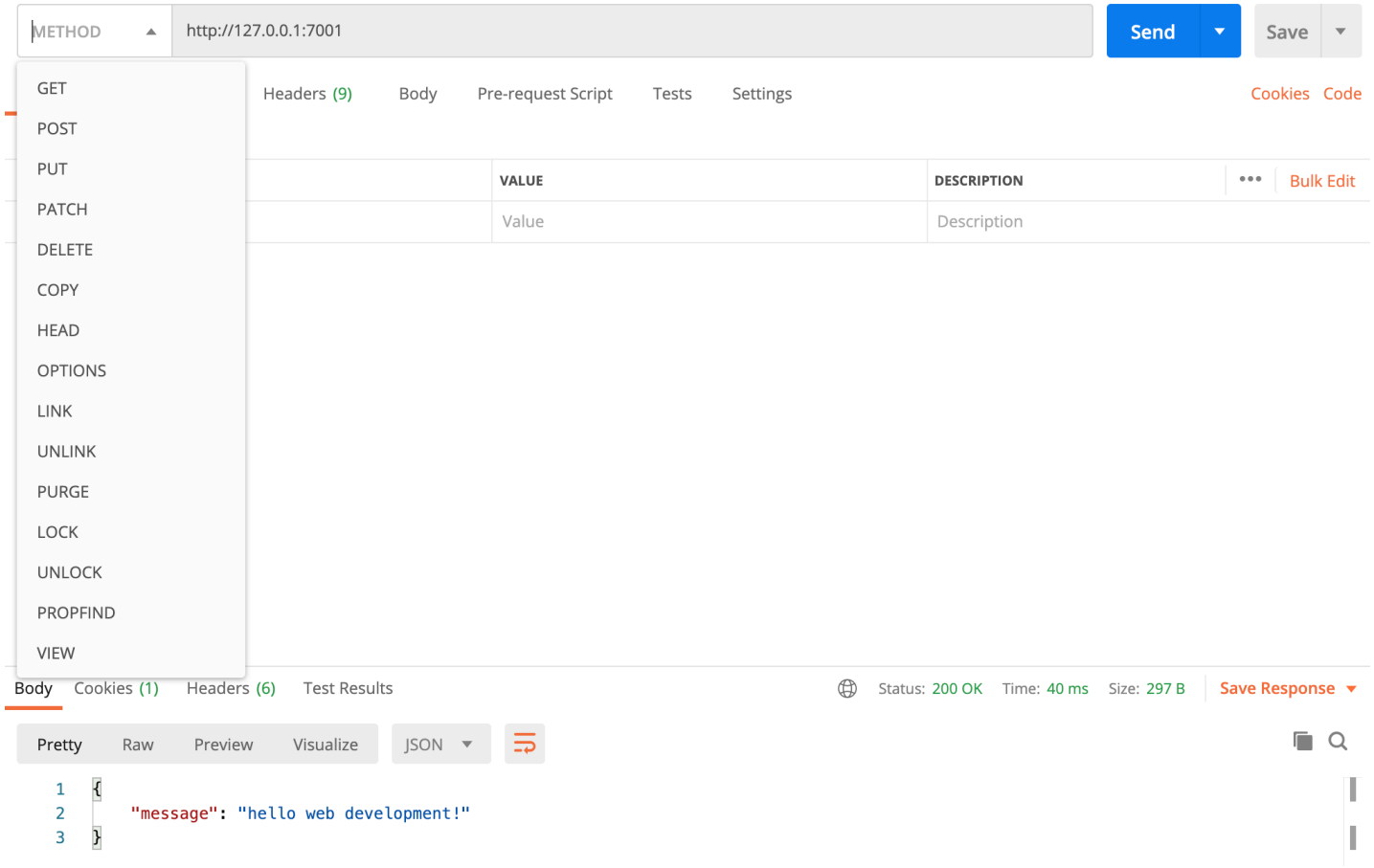
```
setCount((count) => count + 1)
```





高阶函数 调用 回调函数

Postman


METHOD
http://127.0.0.1:7001
Send
Save
GET
POST
PUT
PATCH
DELETE
COPY
HEAD
OPTIONS
LINK
UNLINK
PURGE
LOCK
UNLOCK
PROPFIND
VIEW
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
Code
VALUE
DESCRIPTION
Bulk Edit
Value
Description
Body
Cookies (1)
Headers (6)
Test Results
Status: 200 OK
Time: 40 ms
Size: 297 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "message": "hello web development!"
3 }

谢　谢