设置定时器

console.log("12323");

setTimeout(function () {

  console.log("12323");

}, 1000);

设置计时器

var time = 0;

console.log("12323");

setInterval(function () {

  time += 2;

  console.log(time + "12323");

}, 1000);

清除计时器

var time = 0;

console.log("12323");

var timer = setInterval(function () {

  time += 2;

  console.log(time + "12323");

  if (time > 5) {

    clearInterval(timer);

  }

}, 1000);

输出当前目录

console.log(\_\_dirname);

高级函数 回调函数

function callFunction(fun, name) {

  fun(name);

}

var newName = function (name) {

  console.log(name + "123");

};

callFunction(newName, "hello");

模块化

var counter = function (arr) {

  return "There are" + arr.length + "elelements";

};

module.exports = counter;

var counter = require("./count");

console.log(counter(["ruby", "learn"]));

模块化，键值对

var counter = function (arr) {

  return "There are" + arr.length + "elelements";

};

var adder = function (a, b) {

  return `the sum of ${a + b}`;

};

module.exports.counter = counter;

module.exports.adder = adder;

var name = require("./count");

console.log(name.counter(["ruby", "learn"]));

console.log(name.adder(2, 3));

事件

var events = require("events");

var Person = function () {

  this.name = name;

};

var myEmitter = new events.EventEmitter();

myEmitter.on("someEvent", function (message) {

  console.log(message);

});

myEmitter.emit("someEvent", "the event was emitted");

绑定事件

var events = require("events");

var util = require("util");

var Person = function (name) {

  //定义一个类

  this.name = name;

};

util.inherits(Person, events.EventEmitter); //Person方法继承类

var xiaoming = new Person("xiaoming"); //新建三个对象

var lili = new Person("lili");

var lucy = new Person("lucy");

var person = [xiaoming, lili, lucy];

person.forEach(function (person) {

  person.on("speak", function (message) {

    console.log(person.name + "said " + message);

  });

});

xiaoming.emit("speak", "hi");

lili.emit("speak", "yooo");

lucy.emit("speak", "come");

读写文件异步

var fs = require("fs");

var readme = fs.readFile("readme.txt", "utf8", function (err, data) {

  fs.writeFile("write.txt", data, function (err, data) {

    console.log("writeme has finished writing");

  });

}); //读文件

console.log("finished");

// fs.writeFileSync("write.txt", readme); //写文件 同步执行

删除文件

var fs = require("fs");

fs.unlink("write.txt", function (err) {

  console.log("delete finished");

});

创建目录写入文件

var fs = require("fs");

// fs.unlink("write.txt", function (err) {

//   console.log("delete finished");

// });//异步

//创建目录

// fs.mkdirSync("stuff");

//异步

fs.mkdir("stuff", function () {

  fs.readFile("readme.txt", "utf8", function (err, data) {

    fs.writeFile("./stuff/write.txt", data, function () {

      console.log("read finished");

    });

  });

});

流和管道

ls | grep app | grep js

//将文件作为流处理读文件和写文件

//将文件作为流处理读文件

var fs = require("fs");

var myReadStream = fs.createReadStream(\_\_dirname + "/readme.txt");

var myWriteStream = fs.createWriteStream(\_\_dirname + "/write.txt");

myReadStream.setEncoding("utf8");

var data = "";

myReadStream.on("data", function (chunk) {

  //data += chunk;

  myWriteStream.write(chunk);

});

myReadStream.on("end", function () {

  //console.log(data);

});

管道

myReadStream.pipe(myWriteStream);

搭建web服务器

//搭建服务器

var http = require("http");

var onRequest = function (request, response) {

  console.log("Request:received ");

  response.writeHead(200, { "Content-Type": "text/plain" });

  //   response.write("hello");

  response.end("hello,kkk");

};

var server = http.createServer(onRequest);

server.listen(3000, "127.0.0.1");

console.log("server listening on port 3000");

服务器响应 JSON

var http = require("http");

var onRequest = function (request, response) {

  console.log("Request:received ");

  response.writeHead(200, { "Content-Type": "application/json" });

  //   response.write("hello");

  var myObj = {

    name: "sadadsa",

    job: "13",

    age: 27,

  };

  response.end(JSON.stringify(myObj));

};

var server = http.createServer(onRequest);

server.listen(3000, "127.0.0.1");

console.log("server listening on port 3000");

服务器响应html

var http = require("http");

var fs = require("fs");

var onRequest = function (request, response) {

  console.log("Request:received ");

  response.writeHead(200, { "Content-Type": "text/html" });

  var myStream = fs.createReadStream(\_\_dirname + "/index.html", "utf8");

  //   response.write("hello");

  myStream.pipe(response);

};

var server = http.createServer(onRequest);

server.listen(3000, "127.0.0.1");

console.log("server listening on port 3000");

模块化，将核心部分进行模块放置，入口函数

外面封装一个函数，在入口处调用

var server = require("./server");

server.startServer();

路由

//搭建服务器

var http = require("http");

var fs = require("fs");

function startServer() {

  var onRequest = function (request, response) {

    console.log("Request:received ");

    if (request.url === "/" || request.url === "/home") {

      response.writeHead(200, { "Content-Type": "text/html" });

      fs.createReadStream(\_\_dirname + "/index.html", "utf8").pipe(response);

    } else if (request.url === "/review") {

      response.writeHead(200, { "Content-Type": "text/html" });

      fs.createReadStream(\_\_dirname + "/review.html", "utf8").pipe(response);

    } else if (request.url === "/api/v1/users") {

      response.writeHead(200, { "Content-Type": "application/json" });

      var jsonObj = {

        name: "2000",

      };

      response.end(JSON.stringify(jsonObj));

    } else {

      response.writeHead(200, { "Content-Type": "text/html" });

      fs.createReadStream(\_\_dirname + "/404.html", "utf8").pipe(response);

    }

  };

  var server = http.createServer(onRequest);

  server.listen(3000, "127.0.0.1");

  console.log("server listening on port 3000");

}

exports.startServer = startServer;

模块化：

server.js

var http = require("http");

var fs = require("fs");

function startServer(route, handle) {

  var onRequest = function (request, response) {

    console.log("Request:received ");

    route(handle, request.url, response);

  };

  var server = http.createServer(onRequest);

  server.listen(3000, "127.0.0.1");

  console.log("server listening on port 3000");

}

module.exports.startServer = startServer;

app.js

var server = require("./server");

var router = require("./router");

var handler = require("./handler");

var handle = {}; //空对象

handle["/"] = handler.home; //添加值路径

handle["/home"] = handler.home; //key value 键值对形式

handle["/review"] = handler.review;

handle["/api/records"] = handler.api\_records;

server.startServer(router.route, handle);

router.js

var fs = require("fs");

function route(handle, pathname, response) {

  console.log("Routing a request");

  if (typeof handle[pathname] === "function") {

    handle[pathname](response);

  } else {

    response.writeHead(200, { "Content-Type": "text/html" });

    fs.createReadStream(\_\_dirname + "/404.html", "utf8").pipe(response);

  }

}

module.exports.route = route;

handler.js

var fs = require("fs");

function home(response) {

  response.writeHead(200, { "Content-Type": "text/html" });

  fs.createReadStream(\_\_dirname + "/index.html", "utf8").pipe(response);

}

function review(response) {

  response.writeHead(200, { "Content-Type": "text/html" });

  fs.createReadStream(\_\_dirname + "/review.html", "utf8").pipe(response);

}

function api\_records(response) {

  response.writeHead(200, { "Content-Type": "application/json" });

  var jsonObj = {

    name: "2000",

  };

  response.end(JSON.stringify(jsonObj));

}

module.exports = {

  home: home,

  review: review,

  api\_records: api\_records,

};

打宝，用webpack 并安装npm

使用nodemon文件， nodemon app开始监控