动态网页：

jsp = HTML + JAVA功能强大可靠，适合大型企业级项目

php = php + HTML 简单易用，适合互联网项目

asp.net = HTML + C# 开发全面，适合windows

node.js = HTML + ‘JS’性能好，适合服务器端IO密集型项目，不适合CPU密集型项目

NodeJS是JS一种软件开发平台

NodeJS是由js编写的，前后端通吃

网站：阿里云镜像<https://npm.taobao.org/mirrors/node>

Node运行 get（事件驱动，非阻塞I/O）

JS数据类型

1. 基本、原生、值类型string/number/Boolean/null/undefined
2. 引用类型：es（error/data/math/array/regexp/object），BOM(window/document/location/event/screen…)、DOM(element/attribute…)
3. 用户自定义类型

NodeJS数据类型

1. 基本、原生、值类型string/number/Boolean/null/undefined
2. 引用类型：es（error/data/math/array/regexp/object）
3. 用户自定义类型{}
4. Node.JS第三方对象:www.npmjs.com

变量、常量

运算符

逻辑结构

选择结构

函数对象

常用组件、第三方工具库、框架

实际项目

nodeJS特有的概念——Module模块

一个项目中多个模块，订单模块，用户模块，支付模块

nodeJS按照功能不同，把函数对象保存为不同文件或者目录，这些统称一个模块

NodeJS每个模块都是一个独立的构造函数，为每一个js代码自动添加为

（function（exports, require, module, \_\_filename, \_\_dirname）{

你自己写的代码

}）；

exports:{}用于声明向外部导出自己成员

require：fn用于导入其他的模块，并创建指定模块对象

module:当前模块对象

\_\_filename 绝对文件路径

\_\_dirname 绝对文件夹路径

都能向外界导出自己的内部成员

exports： 请求module.exports导出内部成员

module.exports:实际导出内部成员

每个模块都可使用自己的require()引入另一个模块——底层是创建指定模块的一个对象实例

module.exports.外部使用名 = 内部名；/module.exports = {名：值， 名：值}

var a = require(‘模块路径’); import 模块 as a

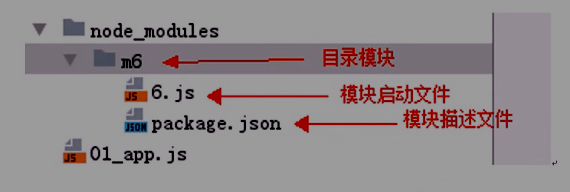
访问值用模块名打点访问

目录模块：

方式一：创建目录m4,其中创建index.js文件，导出需要公开数据，其他模块引入其他模块引入require（./m4）

方式二：创建目录m5，创建5.js，导出需要公开数据，创建package.json,main指定启动5，js，其他模块引入require（./m5）

方式三（主要）：创建目录（文件夹），必须名为node\_modules,在里面创建目录模块，每个模块中需要创建package.json，其中声明main属性指定默认执行启动js文件，js文件中导出需要的公共数据，其他模块引入此模块只需要require(‘模块名’);



原生模块（重点）NodeJS解析引擎自带的

Buffer（缓冲区）

本质为一块内存区域，用于暂存以后要用到数据，为缓存无需引入

let buff = Buffer.alloc(1024);分配指定大小的缓冲区 1024字节 最大为512\* 1024 512KB

let buff = Buffer.from([1, 2, 3])使用一个数组创建缓冲区

let buff = Buffer.from(‘abv’);使用一个字符串创建缓冲区

let str = buff.toString()将一个缓冲区内容转换字符串{如果缓冲区内容为字符串}

QueryString

查询字符串，用于HTTP请求URL中查询字符串，需要引入

let obj = qs.parse(str);把查询字符串解析js对象

let str = qs.stringify(obj)把js对象转换为查询字符串

URL

此模块用于解析一个http请求地址，获取其中各个部分，需要引入

let obj = url.parse(str， true);把一个url地址为js对象，true会把查询字符串转换为js对象

fs

文件：node.js我们系统文件或者目录统称文件

fs模块提供对文件与目录的增删改查读

1. 同步读写文件（阻塞）
   1. var data = fs.readFileSync(‘文件名’); 同步读取文件内容
   2. fs.writeFileSync(文件名.数据)； 同步向文件写内容与py， w相似
   3. fs.appendFileSync(文件名，数据); 同步向文件追加内容
2. 异步读写文件（非阻塞）
   1. fs.readFile(文件名，function（err,文件数据）{}) 异步读取数据,读取完调用fucntion
   2. fs.writeFile(文件名，数据，function(error){}) 异步写入数据,写入完function
   3. fs.appendFile(文件名,data,function(error){}) 异步追加数据,追加完function

http

自己创建web服务器，需要自己引入模块

1. var server = http.createServer(); 创建服务器对象
2. server.listen(3000(端口号)), 绑定监听端口1-65535
3. server.on(“request”,(req, res)=>{}) 接受客户端请求
4. res.setHeader(“Content-Type”,”text/html;charset=utf-8”);
5. res.end(str…);并且响应客户端消息

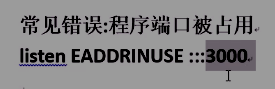
req:请求对象,保存客户端很多信息

let url = req.url得到客户请求的地址

req.method 客户请求方式

req.httpVersion http版本

req.headers 请求头



res：响应对象，保存服务器端向客户发送数据

res.statusCode = 200 响应状态码

res.setHeader() 设置响应头

res.write() 向客户端输入响应消息主体

res.end() 通知客户端响应消息结束只能一次

第三方数据库 用npm下载第三方模块www.npmjs.com

SQL cmd npm install mysql

赋值模块

1、加载mysql模块 const mysql = require(‘mysql’);

2、创建到数据库服务器连接 let conn = mysql.createConnection({host:连接数据,user数据库用户名,password数据库密码，database数据库名称，port端口})

3、发送sql语句并且获取服务器返回结果let uid = 1; sql = ‘xxxxx = ?’; conn.query(sql, [?=数据]，(err,result)=>{})

查询字符串，result自动发回查询到的数组

4、关闭链接 conn.end();

npm i express http高级模块

使用官方提供http模板可以创建一个web服务器应用，但是此模块很底层，推荐使用http高层模块，express是基于node,jshttp模块的高层模块，简化服务器端开发

1. 加载http,express模块
2. 创建express对象，再创建服务器对象，并且绑定监听端口

var app = express();

var server = http.createServer(app);

server.listen(3000);

3、接收客户请求get/index.html app.get(/index.html, (req, res) => {});

4、接收客户请求post/index.html app.post(/index.html, (req, res) => {});

res.sendFile(\_\_dirname+文件绝对路径);

res.send(返回html字符串);

res.json(返回json obj);

1. 接收参数

get/login.do?uid=10&uname=tom

app.get(‘login.do’,(req, res) => {

express为每一个req对象添加属性query

req.query.uid;

req.query.uanme;

});

get/login.do/10/tom

app.get(“/login.do/:uid/:uname”,(req, res) => {

express为内个req对象添加一个params属性

req.params.uid;

req.params.uname;

});

建立连接池

let poor = mysql.createPool({创建mysql连接参数……connectionLimit：5})创建连接池

pool.getConnection((err, conn) => {}) 租用连接池

conn.release() 归还连接池

浏览器发送get请求

url,表单get，ajax.get、超链接、js跳转、img src

浏览器发送post请求

表单post、ajaxpost

浏览器发送delete请求

ajax delete

浏览器发送put请求

ajax put

指定静态目录：（目录下所有资源可以直接通过地址栏访问）

app.use(express.static(‘静态资源目录’));

指定之后不用访问文件夹就可以直接访问资源

npm i body-parser 处理post请求

app.use(bodyParser.urlencoded({extended:false}));

接收数据：req.body.参数名；

npm i cookie-parser 处理 cookie

app.use(cookieParser())

npm i express-session 处理session

app.use(expressSession({

resave: false, 每次请求是否重新设置session

saveUninitialized：true 每次请求是否设置cookie

secret: ‘teducn’https加密传输秘钥

}))

在sesssion中保存数据：req.session.名称=值；

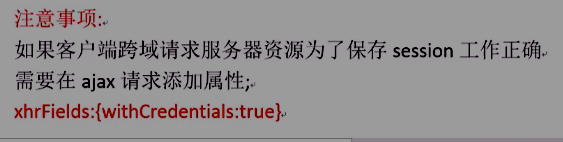
npm i cors 跨域

app.use(cors({

origin:[‘允许跨域的地址’，“xx”“xxxx”],

credentials:true

}))



中间件-路由级中间件

express是一个自身功能极简，完全由中间件构成web框架，从本质上讲是一个express应用就是由中间件组成系统

中间件是一个函数，可以访问请求对象和响应对象，req request, res response可以控制请求-响应流程，有一个参数next变量（下一个中间件。）

中间件功能包括

执行任何代码

修改请求和响应对象

终结请求-响应循环【兰载器】

调用下一个中间件

中间件分类

应用级中间件

路由级中间件

错误处理中间件

内置中间件

标准语法

app.use(url，（req, res, next）=>{})

<url:拦载地址>

req:请求对象

res:相应对象

next：调用下一个中间件或路由

res.redirect(‘url’)重定向

路由中间件 路由：请求方法+请求地址+处理函数

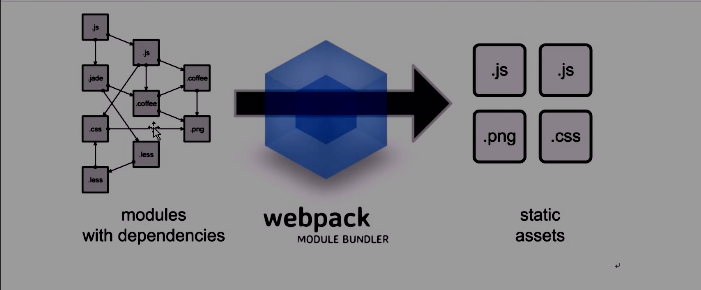
let router = express.Router(); 创建路由中间件对象

router.get(path, (req, res) =>{});

router.post(path, (req,res) =>{});

app.use(‘/user’, router);挂载路由中间件

webpack是一款流行的打包工具，将不同的资源最为模块化处理，生产一个或者一组文件

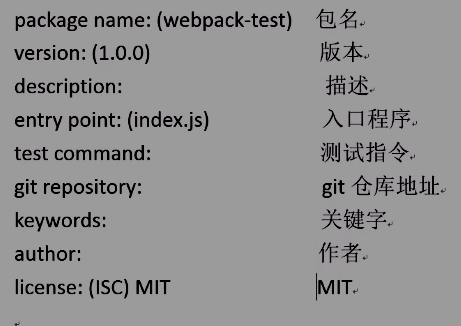


创建目录：mkdir 目录

进入目录：cd 目录

初始化目录： npm init

询问一些问题：



ok

开始安装依赖工具，安装全局webpack

npm i webpack –save –dev



打包js文件：webpack test.js test.bundle.js

打包css文件 webpack test.css test.bundle.js –module-bind’css=style-loader!css-loader’;

安装css加载模块：npm i css-loader /style-laoder –save –dev 

监控项目，实施修改、过程

webpack word,js word.bundle.js –progress

webpack word,js word.bundle.js –watch