First

1.使用云服务器

自己搭建私有服务器：费用昂贵

租用与服务器：费用划算，维护成本低

2.使用新浪云服务器

(1)注册新浪微博账号

(2)使用新浪微博绑定新浪云账号

(3)登录新浪与开通云应用，获得免费域名

(4)管理云应用，上传自己的代码

(5)客户端访问测试

拓展知识：

一级域名：.com .cn 无法购买

二级域名：jd.com tmooc.cn 可以购买

三级域名：mail.jd.com ftp.jd.com 无需购买

3.在新浪云应用中使用数据库

4.使用git来维护代码

版本管理软件：用于记录软件的开发过程，实现团队成员间的协作。实现代码的每一次提交人、提交时间、提交内容——每一次提交成为一个版本。最终实现代码的存储、共享、合并、追踪、回退等功能

常见的版本管理软件：CVS、SVN、Git

版本管理软件的基本概念：

(1)Repository/Repo：仓库，一个小牧所涉及到所有文件以及每一个文件的创建修改过程，称为仓库，即 仓库 = 文件 + 历史

(2)Commit：提交，每一次修改保存入仓库，称为提交

(3)Workspace：工作区，程序员修改代码的地方，就是一个目录

(4)Stage/Cache：缓存区/暂存区/舞台，用于临时存储项目修改的内存区域

(5)Clone：克隆，将远程Git仓库(文件+历史)复制一份到本地——第一次

(6)Push：推送，将本地仓库(文件+历史)复制到远程仓库

(7)Pull：拉取，将远程Git仓库(文件+历史)的最新修改更新到本地

Git常用命令：

(1)git config：配置Git的使用特性

git config --global user.name

git config --global user.email

(2)git init：把当前目录初始化为一个Git仓库——整个项目的第一个仓库，其他同事都克隆此仓库即可

(3)git status：查看当前工作区/暂存区状态，有没有修改/增加/删除

(4)git add：把工作区中的修改添加到暂存区

git add 文件名 添加指定文件的修改

git add . 添加所有的修改

(5)git commit：把暂存区中的修改提交到仓库中，生成一个新的版本

git commit -m '提交的原因说明'

提示：若没有提供-m已经原因说明，会自动进入vi编辑器，让用户输入

(6)git log：查看仓库中的提交日志（谁+时间+提交了什么）

5.使用Git来维护新浪云中的代码

说明：每一个新浪云中的云应用都对应一个Git仓库，可以克隆岛本地，修改提交，最后推送本地仓库中的最新内容到远程仓库

(1)克隆与仓库到本地

git clone 云仓库的URL地址

(2)在本地工作区中修改项目文件

(3)从本地工作区添加修改到暂存区

git add .

(4)从本地暂存区提交到本地从仓库中

git commit -m '...'

(5)将本地仓库中最新历史推送到云仓库

git push 云仓库的URL地址 master

6.使用Git来维护新浪云中的代码——nodejs

(1)克隆云仓库克隆岛本地

git clone 云仓库地址

(2)进入仓库目录，编写Node.js程序

注意：项目必须存在package.json文件，其中必须有：

npm启动

"scripts": {

"start": "node app.js"

},

安装依赖

"dependencies": {

"express": "^4.16.4"

}

新浪云启动Node.js项目默认使用命令：npm i&&npm start

(3)把修改添加到暂存区

(4)把暂存区提交到本地仓库

(5)把本地仓库的更新推送到云仓库

7.使用GitHub

Git：是一款版本管理软件

GitHub.com：是一个网站，提供了免费的源代码托管服务，如果想管理被托管的代码，底层使用Git软件

(1)注册一个邮箱，用于接受验证邮件

(2)在GitHub.com注册一个账户

(3)登录GitHub账户，创建一个应用/仓库

(4)在本地克隆GitHub仓库/添加/提交/推送本地的修改

Second

1.NPM的拓展使用

1.npm init出事话一个node.js项目——创建必须的package.json文件

npm init -y创建必须的package.json文件

2.npm help json查看package.json文件中可用的所有条目

3. npm install xxx：下载并安装指定的扩展依赖模块

npm install xxx --save：下载并安装指定的依赖模块，并声明为“运行时依赖”

npm install xxx --save-dev：下载并安装指定的依赖模块，并声明为“开发时依赖”

npm install: 根据package.json中的“运行时依赖”和“开发时依赖”列表进行指定模块的下载

4. npm run xxx：执行当前目录下package.json中的scripts所定义的命令；若xxx是test/start/stop/rebuild之一，可以省略掉run简写为 npm xxx，即npm rum start等价于npm start

=================================

package.json中的特殊用法：

{

"scripts":{ 定义可以使用npm run执行的命令

"test": "node ./test.js",

"start": "node ./main.js"

},

"dependencies":{ 定义当前项目的“运行时依赖”

"express": "^4.3.0",

"mysql": "^2.16.0"

},

"devDependencies":{ 定义当前项目的“开发时依赖”

"scss"： "^1.15.0",

"gulp": "2.0.0"

}

}

运行时依赖：指当前项目在部署到服务器上供客户端请求时需要使用到的第三方模块

开发时依赖：指当前项目在程序员开发阶段（在最终运行之前）需要使用到的第三方模块

2.模块的使用

注意：在js中使用全局变量/函数，会自动成为全局对象的成员——称为“全局对象的污染问题”；后续重名的变量/函数会覆盖之前的变量/函数。实际开发中，应努力避免使用全局变量/函数

解决方案：——模块(Module)

|  |  |  |  |
| --- | --- | --- | --- |
|  | CommonJS规范 | ES6中的模块规范 | |
| 导出 | //x.js  module.exports = {  age:20;  work:function(){ }  } | //x.js  export var age = 20  export function work (){}  //每一个模块可以到处多个内部成员 | //x.js  export default {  age:20,  work:function(){ }  //每一个模块只能到处一个默认对象 |
| 导入 | //y.js  const obj = require('./x')  console.log(obj.age)  obj.work | //y.js  import (age,work) from './x'  console.log(age)  work() | //y.js  import obj from './x'  console.log(obj.age)  obj.work() |
| 书写位置 | 导入/导出可以写在任意位置 | 导入/导出只能写在最顶级 | |
| 适用场合 | 服务器端nodejs中使用，浏览器中不能使用 | 服务器端nodejs不支持——这是客户端浏览器中的js模块规范——目前所有浏览器都不能直接支持此规范 | |

3.Webpack的使用

官网：webpackjs.com

作用：把多个静态资源文件(脚本/样式/图片)，“打包”为一个/少数几个今天资源文件——减少客户端请求次数/数据传输量，提高web访问效率

使用方法：先使用传统方法学完一个web项目(若干js/css/图片...)，再使用webpack把静态资源打包，在HTML中引入打包后的资源文件让客户端请求

使用步骤：

0.按照传统方法编写web项目

1.下载并安装webpack所需要的所有模块

npm install webpack --save-dev

2.在项目根目录下创建webpack主配置文件

webpack.config.js 注意文件位置/名称

3.运行webpack，读取配置文件，执行打包工作

node ./node\_modules/webpack/bin/webpack.js

4.修改html文件，只引入打包后得到的打包文件即可