**NodeJS**

1.Nodejs简介

# 1.1 什么是JavaScript？

# 接触过前端的同学都知道，javascript是一门运行在浏览器端的脚本语言，用来做客户端页面的交互。

# 1.2 JavaScript的运行环境呢？

# 故名思意他的运行环境就是浏览器，但是真的是这样的吗？其实不然，他的真正运行环境其实是运行在浏览器内核中的js引擎。为什么是这样的了？因为通俗来讲浏览器得作用就是用来浏览网页得，我们在浏览器除了执行js外还能执行其他得东西，比如我们在浏览器输入一个地址敲回车就可以发送请求并且接收服务器得响应。所以说浏览器的第一功能是请求一个http地址，也就是可以封装一个请求报文出来，将一个url的地址封装成一个请求报文，这个报文到服务端，然后给我们一个响应报文，然后在将响应解析出来，这也是浏览器的最大的作用。当然服务器响应的内容有可能不一样，比如说服务器返回一个html文件，css文件，img文件，用来渲染，我们称之为渲染引擎，除此之外，还可以执行js，由js引擎来完成，因此可以说javascript的运行环境是浏览器中的js引擎，而不是浏览器。浏览器是一个大的概念。

# 浏览器的javascript可以做什么？

# 1.3 浏览器中的javascript可以做什么

# 之前由提到过，javascrip就是用来做交互的，但是笼统的说交互有点不明确，具体一点可以分为：“

# （1）操作dom（也就是对dom的增删改，注册事件之类的事情）

# （2）发送AJAX请求/跨域

# （3）BOM交互如给我们提供页面跳转，历史记录，控制台打印日志

# （4）ecmascript(js的核心语言，如用来定义变量，函数等)

# 1.4、浏览器端的javasscript不可以做什么？

# 通过之前的了解，貌似javascrip对我们日常的交互都可以完成，貌似什么都可以做，但是他也有不能做的事。如：

# （1）涉及到端对端的应用程序，我们需要操作文件，浏览器中的javascript是不能进行文件操作的，虽然h5里面提供了关于文件相关的API，但是这些API大多数只限于只读的层面 ，不能像传统的语言如java,通过传一个路径，然后将对应的文件读出来，说白了就是不能进行文件和文件夹的CURD(增删查改).

# （2）浏览器端的javascript也没有办法去操作操作系统，如获取操作系统的版本之类的。

# 那么为什么不能进行这两类操作？其实是出于安全考虑，因为，js这门语言运行的环境比较特殊，说他特殊，特殊在什么地方？虽然我们编写好的js代码最终会放在服务器上，但是他毕竟不是在服务器上执行的，而是通过服务器发送到浏览器端执行的，在浏览器端执行文件的操作显然是不安全的，因此说这些功能在客户端不是不能做，而是由于特殊的运行环境没法做

# 1.5、了解了客户端的js所能做的事，我们必须还要明确一个概念，那就是javascrip只能运行在浏览器端吗？

# 前端开发人员都知道，javascrip是有ecmascrip语言，BOM，DOM组成的，在语言层面，她只是给我们提供一些操作语法，如定义变量，函数，类型，流程控制等的操作。而BOM，DOM是浏览器提供的，并非es提供的。因此我们常提及的js其实就是es，js的大部分功能（DOm，BOM（浏览器开放出来的API）等的操作）都是由 浏览器的执行引擎决定的 ，这也衍生出一个观点，任何一门编程语言 ，他的能力不是由语言本身决定的，而是由他的执行环境决定的。比如说java，他即是一门语言也是一个平台，对于javascript来说语言就是es，平台是浏览器。那么js只能运行在浏览器中吗？非也！对于大多数语言，都是运行 在一个平台上的，比如java只运行在虚拟机上，但是也有运行在多个平台的语言, java在一定层面上来讲是没有必要运行在多个平台上的，因为虚拟机是跨平台（也就是跨操作系统如window，linux等）的。js同样是可以运行在多个平台的，浏览器之所以能过运行js，是因为他由js的执行引擎。js同样，只要由支持他的平台就可以执行。因此说要想语言有很强大的功能，只需要提供强大的平台，node就是这样一个平台，能够执行js，那么node到底是什么东西？接下来简述node是个什么鬼

# 什么是node？

根据官方文档可以知道，node就是一个给予谷歌v8引擎的一个javascript的运行时的平台，可以理解为运行js的一个虚拟机。他使用的是一个事件驱动，非阻塞I/O模型 ，他是将js的运行环境搬到了服务器端，和客户端没有一点关系。是一个纯服务端的东西，node只是为js提供了一个平台。node里面其实还分了两块，一是封装了v8引擎，目的是为了执行es（如定义变量，定义函数等），另外一个提供了大量的工具库,是帮助node实现各种功能的，提供了一些以前js的环境办不到的事情，比如文件操作，网络操作，操作系统的操作。

    既然node是一个平台（所谓的平台就是用来运行特定语言的），也就意味着node是用来运行语言的，那么java也是语言，node能运行java吗？据nodejs创始人Ryan Dahl回忆，他最初是选择了Ruby这门语言，但是Ruby这门语言的虚拟机效率不怎么样最终放弃了，按照这种思路，貌似node将java的虚拟机集成进来应该可以运行java，但node作者最终选择了javascript。这样js就实现了在服务端运行的可能，js运行在node平台上（分为v8部分，用来执行es，和大量的工具库组件（API）称之为libuv，提供了以前js的环境办不到的事，如文件操作，网络操作等等）。

知道了什么是node,应该还要清楚node在web中有什么用途？

    （1）node可以接受客户端用户的所有请求，并且能够快速的给出响应，因此node可以用来做网站。

    （2）node可以作为一个中间层来来分发调用数据接口，比如有一个网站数据是有java提供的，我们可以让node作为一个中间层，来接受用户的请求，然后通过node来调用java数据接口，获取到数据后直接在node层面做html的封装，然后将渲染好的页面直接给用户。为什么要这样做，直接请求java接口不行吗，这是因为node被称之为高性能的web服务器，在并发和抗压方面都比传统的平台要好很多，因此这样一包装可以极大的减轻服务器的开发。

    通过上面的两点，可以总结出，node在web中要么从前端页面到后端服务全包了，一个是只做其中的一点。

    一言以蔽之，node就是一个javascript的运行环境（平台），他不是一门语言，也不是javascript的框架。可以用来开发服务端应用程序，web系统。其特点是体积小，快速，高性能。

简单的说 Node.js 就是运行在服务端的 JavaScript。

Node.js 是一个基于Chrome JavaScript 运行时建立的一个平台。

Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。

## 2.node环境的安装

中文官网

http://nodejs.cn

## C:\Users\ADMINI~1.SKY\AppData\Local\Temp\enhtmlclip\Image(12).png

我们可以使用以下命令来查看当前的 Node 版本：

$ node -v

v4.4.3

**注意：**不同版本间可能是有差异的。

**3.** 第一个Node.js程序：Hello World！

### 脚本模式

以下是我们的第一个Node.js程序：

创建helloworld.js文件

写入内容然后保存

console.log("hello world");

## 保存该文件，文件名为 helloworld.js， 并通过 node命令来执行：

node helloworld.js

程序执行后，正常的话，就会在终端输出 Hello World。

通过http模块创建服务器

如果我们使用PHP来编写后端的代码时，需要Apache 或者 Nginx 的HTTP 服务器，来处理客户端的请求相应。不过对 Node.js 来说，概念完全不一样了。使用 Node.js 时，我们不仅仅在实现一个应用，同时还实现了整个 HTTP 服务器。

1、引入http模块

var http = require("http");

2、创建服务器 接下来我们使用 http.createServer() 方法创建服务器，并使用 listen 方法绑定 8888 端口。 函数通过 request, response 参数来接收和响应数据。

        //引用模块

        var http = require('http');

        //创建一个服务器，回调函数表示接收到请求之后做的事情

        http.createServer(function(req,res){

            res.writeHead(200,{"Content-Type":"text/html;charset=utf-8"});

            res.write("呵呵");

            res.end("你好，隔壁的老王");

        //监听端口

        }).listen(8888);

        console.log("hahaha");

浏览器运行：127.0.0.1：8888

呵呵你好，隔壁的老王

你会发现，我们本地写一个js，打死都不能直接拖入浏览器运行，但是有了node，我们任何一个js文件，都可以通过node来运行。也就是说，node就是一个js的执行环境。

4.**Nodejs**中的模块化

JavaScript是一个强大面向对象语言，它有很多快速高效的解释器。然而， JavaScript标准定义的API是为了构建基于浏览器的应用程序。并没有制定一个用于更广泛的应用程序的标准库。

CommonJS规范的提出,主要是为了弥补当前JavaScript没有标准的缺陷。它的终极目标就是：提供一个类似Python，Ruby和Java语言的标准库,而不只是停留在小脚本程序的阶段。

用CommonJS API编写出的应用，不仅可以利用 JavaScript开发客户端应用，而且还可以编写以下应用。

  •服务器端JavaScript应用程序。（nodejs）

  •命令行工具。

  •桌面图形界面应用程序。

  CommonJS就是模块化的标准，nodejs就是CommonJS（模块化）的实现。

Node应用由模块组成，采用CommonJS模块规范。

在Node中，模块分为两类:

一类是Node提供的模块,称为核心模块；另一类是用户编写的模块，称为文件模块。

• 核心模块部分在Node源代码的编译过程中，编译进了二进制执行文件。在Node进程启动时，部分核心模块就被直接加载进内存中，所以这部分核心模块引入时，文件定位和编译执行这两个步骤可以省略掉，并且在路径分析中优先判断，所以它的加载速度是最快的。如：HTTP模块 、URL模块、Fs模块都是nodejs内置的核心模块，可以直接引入使用。

• 文件模块则是在运行时动态加载，需要完整的路径分析、文件定位、编译执行过程、速度相比核心模块稍微慢一些，但是用的非常多。这些模块需要我们自己定义。接下来我们看一下nodejs中的自定义模块。

CommonJS（Nodejs）中自定义模块的规定：

1.我们可以把公共的功能抽离成为一个单独的 js 文件作为一个模块，默认情况下面这个模块里面的方法或者属性，外面是没法访问的。如果要让外部可以访问模块里面的方法或者属性，就必须在模块里面通过 exports 或者module.exports暴露属性或者方法。

2. 在需要使用这些模块的文件中，通过require的方式引入这个模块。这个时候就可以使用模块里面暴露的属性和方法。

demo1:

config.js文件

let str = "我就是隔壁老王";

let str1 = "我是隔壁的大黄";

exports.str = str;

module.exports.str1 = str1;

function fun(){

    console.log("heihei");

}

exports.fun = fun;

Commonjs.js文件

var http = require('http');

var config = require("./config.js");

var server = http.createServer(function(req,res){

    res.writeHead(200,{"Content-Type":"text/html;charset=utf-8"});

    res.write("hahahahah");

    res.write(config.str);

    res.end();

});

server.listen(6666,"127.0.0.1");

demo2:

tools.js

var tools = {

    \_add:function(a,b){

        return a + b;

    }

}

exports.\_add = tools.\_add;

config.js

var tools = require("./tools.js");

console.log(tools.\_add(1,2));

ES6的模块化:

现在ES6自带了模块化， 也是JS第一次支持module， 在很久以后 ，我们可以直接作用import和export在浏览器中导入和导出各个模块了， 一个js文件代表一个js模块；

现代浏览器对模块(module)支持程度不同， 目前都是使用babelJS， 或者Traceur把ES6代码转化为兼容ES5版本的js代码;

前端模块化

1. 模块化的好处

前后端分离以来，经常提到前端工程化，前端工程化是一个高层次的思想,前端工程化主要包括四个方面：模块化、组件化、规范化、自动化。而模块化和组件化是为工程化思想下相对较具体的开发方式,因此可以简单的认为模块化和组件化是工程化的表现形式。

模块化带来的好处：

   1 避免变量污染，命名冲突

   2 提高代码复用率

   3 提高维护性

   4 依赖关系的管理

2. 什么是模块化？

那具体什么是模块化呢,还是举简单的例子,我们要写一个实现A功能的JS代码,这个功能在项目其他位置也需要用到,那么我们就可以把这个功能看成一个模块采用一定的方式进行模块化编写,既能实现复用还可以分而治之,同理在写样式的时候,如果我们需要某种特殊的样式,会在很多地方应用,那么我们也可以采用一定的方式进行CSS的模块化,具体说来,JS模块化方案很多有AMD/CommonJS/UMD/ES6 Module等,CSS模块化开发大多是在less、sass、stylus等预处理器的import/mixin特性支持下实现的,具体技术大家自行学习。

3. ES6的模块化

现在ES6自带了模块化， 也是JS第一次支持module，我们可以直接作用import和export在浏览器中导入和导出各个模块了， 一个js文件代表一个js模块；

ES6的模块化的基本规则或特点：

　　　　1：每一个模块只加载一次,每一个JS只执行一次,如果下次再去加载同目录下同文件,直接从内存中读取。

　　　　2：每一个模块内声明的变量都是局部变量， 不会污染全局作用域；

　　　　3：模块内部的变量或者函数可以通过export导出；

　　　　4：一个模块可以导入别的模块

　　　　5：ES6的模块自动采用严格模式

　　　　6：模块顶级作用域的this值为undefined

　　　　7：需要外部使用的部分，必须导出才可以

使用：

// 导出 test1.js

export let a = '1'

export function sum(num1,num2){

    return num1+num2

}

// 引入 test2.js

import {a} from './test1.js';

import {sum} from './test1.js';

console.log(a);

console.log(sum(2,3));

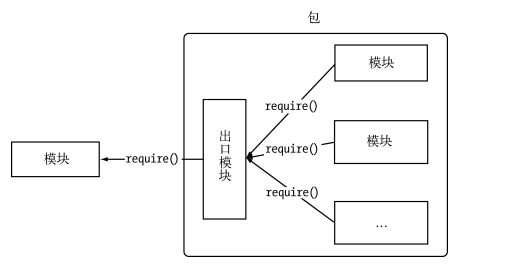
//xxx.html使用

<script src="js/test2.js" type="module"></script>

5. 包与**NPM**

1. 包

Nodejs中除了它自己提供的核心模块外，我们可以自定义模块，也可以使用第三方的模块。Nodejs中第三方模块由包组成，可以通过包来对一组具有相互依赖关系的模块进行统一管理。



完全符合CommonJs规范的包目录一般包含如下这些文件。

• package.json :包描述文件。

• bin  :用于存放可执行二进制文件的目录。

• lib :用于存放JavaScript代码的目录。

• doc :用于存放文档的目录。

在NodeJs中通过NPM命令来下载第三方的模块（包）。

https://www.npmjs.com/package/silly-datetime

命令:

npm i silly-datetime

文件:

var date = require('silly-datetime');

var sd = date.format('new Date',"YYYY-MM-DD hh:mm");

console.log(sd);

2.npm介绍

npm是世界上最大的开放源代码的生态系统。我们可以通过npm下载各种各样的包，这些源代码（包）我们可以在https://www.npmjs.com找到。

npm是随同NodeJS一起安装的包管理工具，能解决NodeJS代码部署上的很多问题，常见的使用场景有以下几种：

允许用户从NPM服务器下载别人编写的第三方包到本地使用。(silly-datetime)

允许用户从NPM服务器下载并安装别人编写的命令行程序(工具)到本地使用。（supervisor）

允许用户将自己编写的包或命令行程序上传到NPM服务器供别人使用。

3.npm命令详解

1. npm -v 查看npm版本

2. 使用 npm 命令安装模块

npm install Module      Name 如安装jq模块：      npm install jquery

3. npm uninstall moudleName 卸载模块

npm uninstall      ModuleName

4. npm list 查看当前目录下已安装的node包

npm

5. npm info jquery 查看jquery的版本

npm info 模块 //查看模块的版本

Node自动重启工具 nodemon

为什么要使用

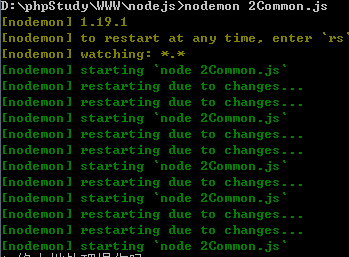
在编写调试Node.js项目，修改代码后，需要频繁的手动close掉，然后再重新启动，非常繁琐。现在，我们可以使用nodemon这个工具，它的作用是监听代码文件的变动，当代码改变之后，自动重启。

如何使用 nodemon

npm install -g  nodemon

安装好后直接使用

nodemon app.js



6.fs模块 文件与目录操作

var fs = require('fs');

//1. fs.stat  检测是文件还是目录

// fs.stat('dir',function(err,err){

//     console.log(stat.isFile());

//     console.log(stat.isDirectory());

// });

// fs.stat('index.html',function(err,stat){

//     console.log(stat.isFile());

//     console.log(stat.isDirectory());

// });

//2. fs.mkdir  创建目录

// fs.mkdir("mycss",function(err,stat){

//     console.log('创建目录成功');

// });

//3. fs.writeFile  创建写入文件

//参数 文件 内容  回调函数

// fs.writeFile("1.txt","我是隔壁的泰山",function(err,stat){

//

// });

//4.fs.appendFile 追加文件

// fs.appendFile("1.txt","我是隔壁的老王",function(err,stat){

//

// });

//5.fs.readFile 读取文件

fs.readFile("1.txt",function(err,data){

    console.log(data);

});

//6. fs.rmdir  删除目录

fs.rmdir('mycss',function(){});

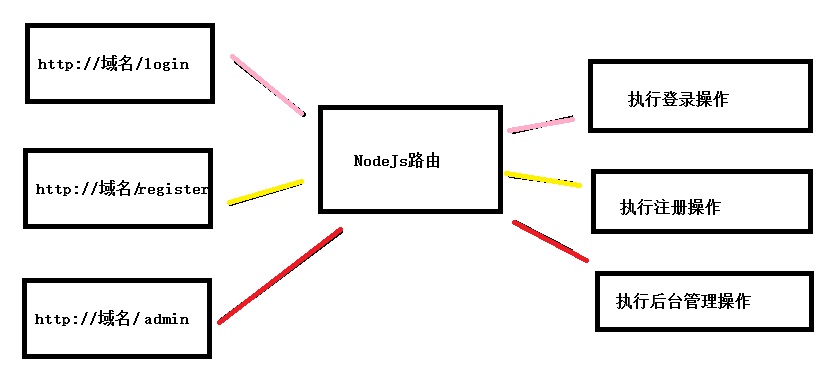
//7.fs.unlink删除文件

fs.unlink('index.html',function(){});

7.路由

官方解释： 路由（Routing）是由一个 URI（或者叫路径）和一个特定的 HTTP 方法（GET、POST 等）组成的，涉及到应用如何响应客户端对某个网站节点的访问。

非官方解释： 路由指的就是针对不同请求的URL，处理不同的业务逻辑。



var http = require('http');

        http.createServer(function(req,res){

            res.writeHead(200,{"Content-Type":"text/html;charset=utf-8"});

            console.log(req.url);

            if(req.url == "/login"){

                console.log("登录");

            }else if(req.url == "/reg"){

                console.log("注册");

            }

            res.end("heihei");

        }).listen(8866);

作业:

1.整理笔记,课堂案例