Node.js，主讲：汤小洋

一、Node.js简介

1. 关于JavaScript

什么是JavaScript?

- JavaScript是一门脚本语言
- 一般运行在浏览器中，用来做客户端的页面交互

JavaScript的运行环境?

- 一般我们说在浏览器中运行JavaScript
- 准确的说，是由浏览器内核中的JS引擎运行JavaScript
- 浏览器的功能，主要有两个：

  1.请求一个HTTP地址：将一个URL地址封装成一个请求报文

  2.解析服务器返回的响应报文：渲染HTML（渲染引擎）、渲染CSS、渲染IMAGE、执行JavaScript（JS引擎）

浏览器中的JavaScript可以做什么?

- DOM操作（DOM元素的获取和更新、注册事件等）
- BOM操作（页面跳转、历史记录、弹窗等）
- ECMAScript
- AJAX/跨域

浏览器中的JavaScript不可以做什么?

- 文件操作
- 数据库操作
- 获取操作系统信息

JavaScript只能运行在浏览器中吗?

- 不是，只要有JS引擎就可以运行JavaScript
- Node.js中就包含了一个JS引擎，可以运行JavaScript

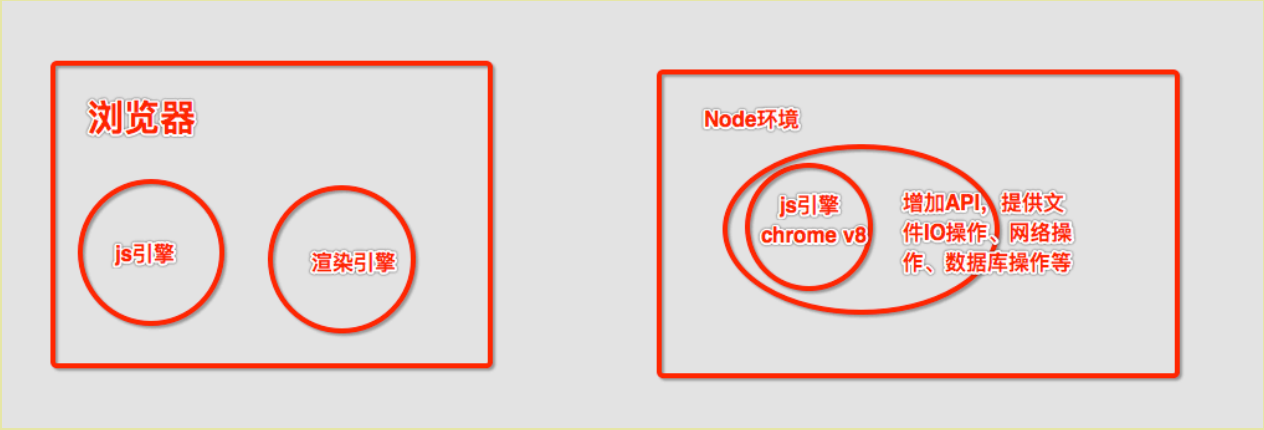
2. 关于Node.js

Node.js（简称为Node）不是一门语言，也不是一个JavaScript框架，也没有一个叫node.js的文件

Node.js 是一个可以让JavaScript在服务器端运行的系统环境，即运行JavaScript不再依赖于浏览器

Node.js 是一个基于Chrome的JS引擎（V8）所开发的软件程序，可以执行ECMAScript，但不支持DOM和BOM操作

Node 提供大量API工具库，使JavaScript可以完成更多功能，如IO操作、数据库操作、获取系统资源等。



3. Node.js作用

开发Web应用程序

- 开发动态网站
- 开发提供数据的服务端API

前端开发工具的基础

- 许多前端开发工具都是基于Node平台

二、Node环境搭建

1. 安装Node

node官网：https://nodejs.org

node中文网：http://nodejs.cn

直接执行软件包进行安装即可，安装完成后打开控制台（cmd），执行命令 `node -v` 查看安装版本即可

2. REPL环境

Node.js提供了一个交互式运行环境，即REPL，在命令行窗口中，输入 `node` 命令并按下回车键，即可进入REPL运行环境

在REPL运行环境中，我们可以完成：JavaScript代码的输入、解释、执行、结果输出等操作。

退出REPL环境：连续按两下 `ctrl+c`

3. 基本使用

在命令行中执行.js文件： `node js文件路径`

示例：

- 处理HTTP请求并响应
- 用户登陆

三、模块化

1. 简介

模块化是指将一个复杂的项目按照一定的规则拆分成一个个模块，进行组合使用，完成项目的开发

- 模块的内部对使用者来说是透明的，在模块内部会进行成员的导出，向外暴露接口
- 使用者在使用时加载对应的模块，并调用模块提供的功能即可
- 方便代码的分层开发，保证每个功能模块的职能单一
- 项目开发时会使用到许多模块，模块间可能会存在依赖关系

Node使用模块化的方式组织代码结构：

- 一个模块可以是一个独立文件，也可以是一个文件夹，但都需要符合特定的要求
- 模块中提供的数据和功能，需要进行导出才能被外部所使用

2. 定义模块并导出

2.1 文件模块

独立文件模块有三种形式：

- `.js` 文件模块，使用js语法方式进行定义，并通过 `module.exports` 或者 `exports` 进行模块功能导出
- `.json` 文件模块，使用json对象定义数据，当通过 `require` 导入模块时，会自动导出定义的json数据
- `.node` 文件模块，使用node.js编译后的二进制文件，不能自行定义

在Node.js环境中使用 `module.exports` 或 `exports` 进行模块的导出：

- module 与 exports 都是Node.js中的全局对象
- module对象

  Node内部提供一个Module构造函数，属性如下：

```
Module {
```

```
Module {
  id: '.',  // 模块的标识符，主模块的id属性值为“.”，其他模块的id属性值为该模块文件的绝对路径
  exports: {}, // 模块的导出对象，为require加载时提供返回结果
  parent: null, // 当前模块的父模块
  filename: '', // 当前模块的文件名称和存储路径
  loaded: false, // 表示模块是否加载完成
  children: [], // 当前模块的子模块
  paths:[] // 加载模块文件的路径顺序
}
```

所有模块都是Module的实例，载入一个模块就是构建一个Module实例

- exports对象

  该对象实际上就是 module.exports 对象，用来简单模块导出操作

2.2 文件夹模块

文件夹模块，要求根目录下必须存在 `index.js` 或 `package.json` 文件，即必须有入口文件

关于package.json文件，格式如下：

```
{
  "name": "模块名",
  "version": "模块版本",
  "description": "模块描述",
  "main": "模块的入口文件",
  "scripts": {
    "命令名": "模块的执行命令"
  },
  "keywords": ["模块的关键字"],
  "author": "模块的作者",
  "license": "模块的开源协议，如：ISC、"
}
```

在Node.js环境中，存在 `package.json` 文件的文件夹，其实是一个独立的Node项目，该文件用于定义项目的相关信息

- 可以使用 `npm` (node package manager) 命令，自动生成package.json文件

  语法： `npm init` 或 `npm init --yes`

3. 导入模块并使用

在Node.js环境中使用 `require(模块路径|模块名)` 进行模块的加载导入：

- 省略后缀名时模块的查找顺序： .js ——> .json ——> .node

- 加载模块时的查找顺序：核心模块——>第三方模块——>用户定义模块

模块的分类：

- 核心模块，在安装完Node环境后，Node环境自带的模块，可直接导入使用
- 第三方模块，使用 `npm` 命令从外网下载并安装的模块
- 用户定义模块，开发者通过模块定义语法自定义的模块

四、第三方模块的安装

1. npm简介

全称node package manager（Node包管理工具）

npm是随同Node.js一起安装的包管理工具，用来安装、卸载、更新软件包/模块等，同时能够解决软件包之间的依赖关系

官网：https://www.npmjs.com

2. npm用法

命令	作用	备注
`npm init [--yes]`	初始化项目/模块	在项目根目录下会生成package.json
`npm install 模块名`	下载并安装模块	切换到项目根目录下执行，下载后的模块文件存储在 `node_modules` 文件夹中
`npm install 模块名@版本号`	下载并安装指定版本的模块	默认安装模块的最新版本
`npm install 模块名 --save|-S`	下载并以生产依赖的方式安装	将模块信息添加到package.json中的dependencies（生产依赖）
`npm install 模块名 --save-dev|-D`	下载并以开发依赖的方式安装	将模块信息添加到package.json中的devDependencies（开发依赖）
`npm install 模块名 -g`	全局安装模块	为系统提供全局功能，存储于node的全局环境中，如 less
`npm install`	自动根据package.json文件中的依赖信息下载包	自动读取package.json文件中的依赖信息并下载安装
`npm uninstall 模块名`	卸载项目中的模块	--save|-S 卸载时删除package.json中的生产依赖记录 --save-dev|-D 卸载时删除package.json中的开发依赖记录 -g 卸载全局模块

`npm info 模块名`	查看模块的描述信息	模块版本、依赖等
`npm update 模块名`	更新模块	--save|-S、--save-dev|-D、-g
`npm run 命令名`	执行指定的命令	执行 `package.json` 文件中的 `scripts` 属性的指定命令

3. 全局模块nrm

该模块为node的全局扩展模块，提供系统命令，用于实现快速 npm 源地址切换

执行 `npm install nrm -g` 进行全局安装

- nrm ls 看地址列表
- nrm use 切换地址
- nrm test 测试速度

4. 模块的安装和使用

模块查找

模块安装

模块使用