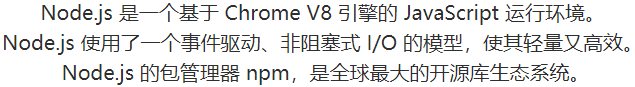
# 进击Node.js基础（一）

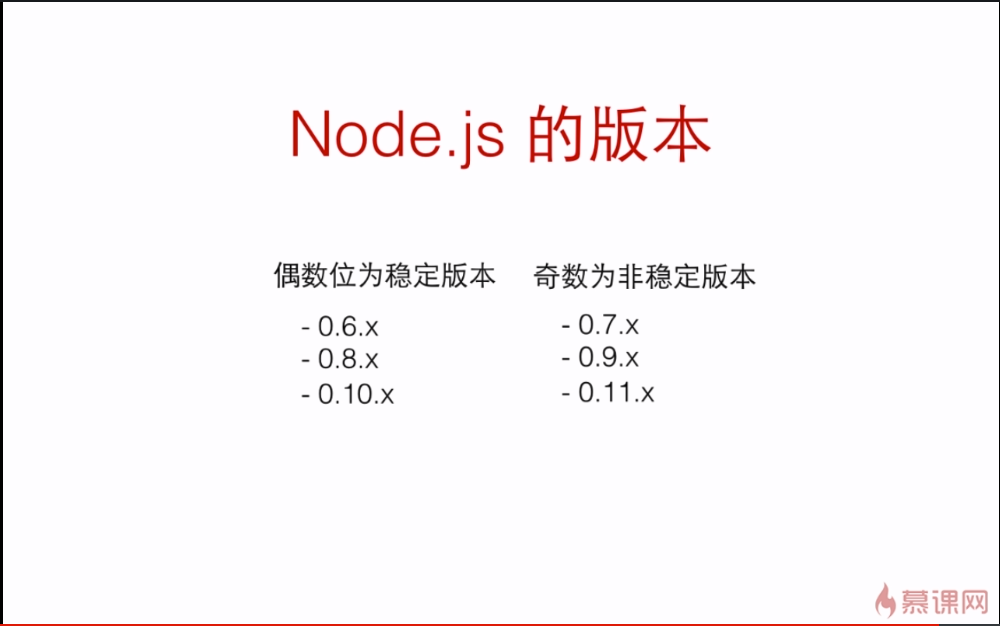
1. Nodejs是JavaScript的运行环境；



1. 基于Nodejs 的技术和完成的项目：



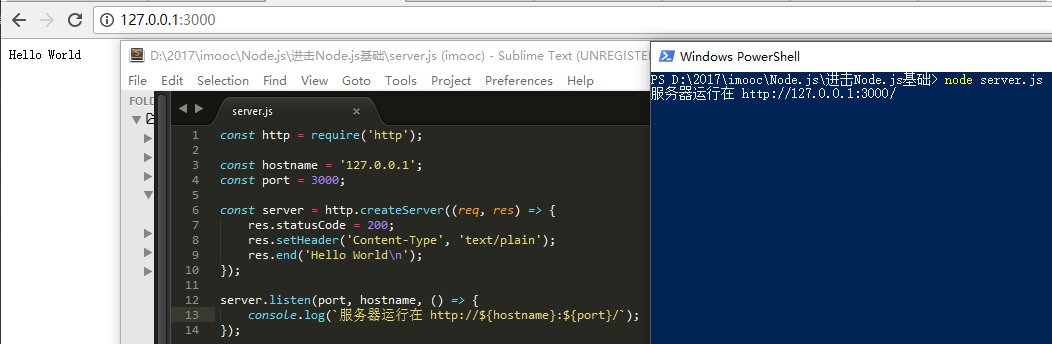
1. 网站推荐：<https://nodejs.org> ；<https://www.npmjs.com> ；<https://stackoverflow.com/> ；<https://github.com/> ；
2. Nodejs版本问题



1. 安装gitbash、安装nodejs；官网直接下载傻瓜式安装；查看版本号：node -v，npm-v
2. 事件冒泡：只要是能够触发事件冒泡的情况下，就考虑取消事件冒泡，或者直接利用事件冒泡



1. 实现一个简单web服务器



1. 浏览器执行环境中存在window全局变量，nodejs执行环境中存在process模块，这是两者的区别之一；
2. 大量js文件在页面引入时，很容易出现变量/方法被覆盖掉，特别是这些文件存在依赖关系的时候；容易导致页面出错；

因为javascript天生缺少一种模块管理机制 来隔离实现不同功能的JS片段；避免他们相互污染；

为此我们经常采用命名空间的方法，把变量和函数限制在某个特定的作用域内；人肉约定一套命名规范来约束代码，从而保证代码的安全执行。如jQuery必须通过$来调用其中的方法；

在commonjs中一个文件就是一个模块,就是一个作用域；

Node js借鉴了Commonjs的模块组织理念；并基于commonjs实现了一套模块管理系统；在Nodejs中每个js文件可以看做是一个独立的模块； 在里边不需要有命名空间，不用担心变量污染，整个文件中的代码可以非常自然的组织起来， 整个模块可以通过彼此的依赖引入组合起来从而形成更强大的功能/功能包； 安装nodejs时会安装npm这个包管理工具，通过该工具可以往项目中引入各种模块，这些模块的像是相互独立的 完整的；

Commonjs规范包括模块、包、二进制、控制台、编码等规范，来约定Javascript怎么来组织编写；

同时大部分标准也是在拟定和讨论之中的；

把不同功能的代码看做是独立的模块，每个模块看做是独立的作用域；

但并不是孤立的，可能存在依赖关系；

每个模块可以分为三个关键的部分，分别是模块的定义，标识，引用；

Commonjs是一套规范呐！

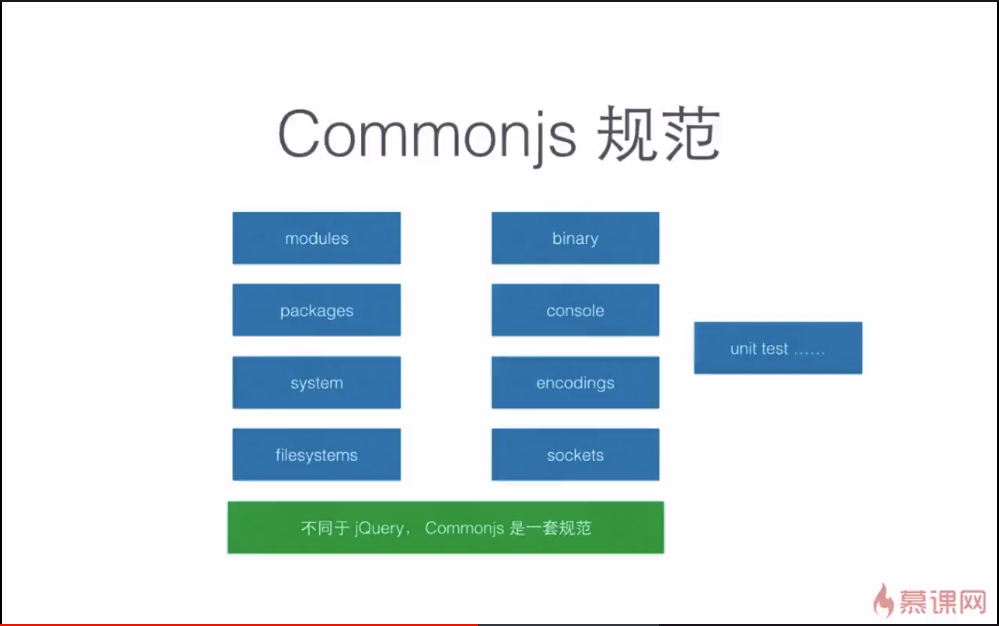
CommonJS模块的特点如下：

所有代码都运行在模块作用域，不会污染全局作用域。

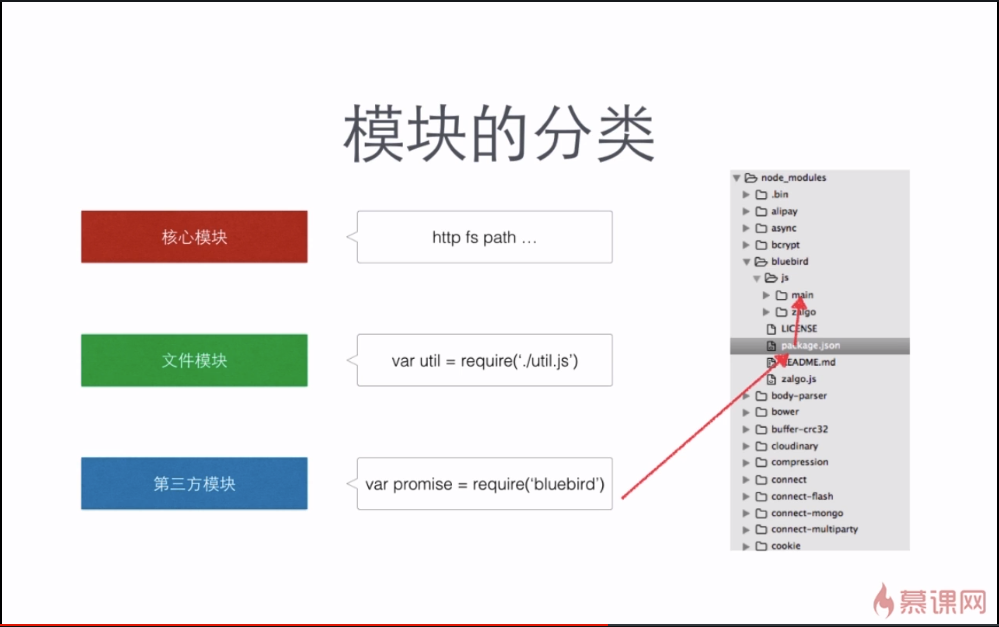
模块可以多次加载，但是只会在第一次加载时运行一次，然后运行结果就被缓存了，以后再加载，就

直接读取缓存结果。要想让模块再次运行，必须清除缓存。

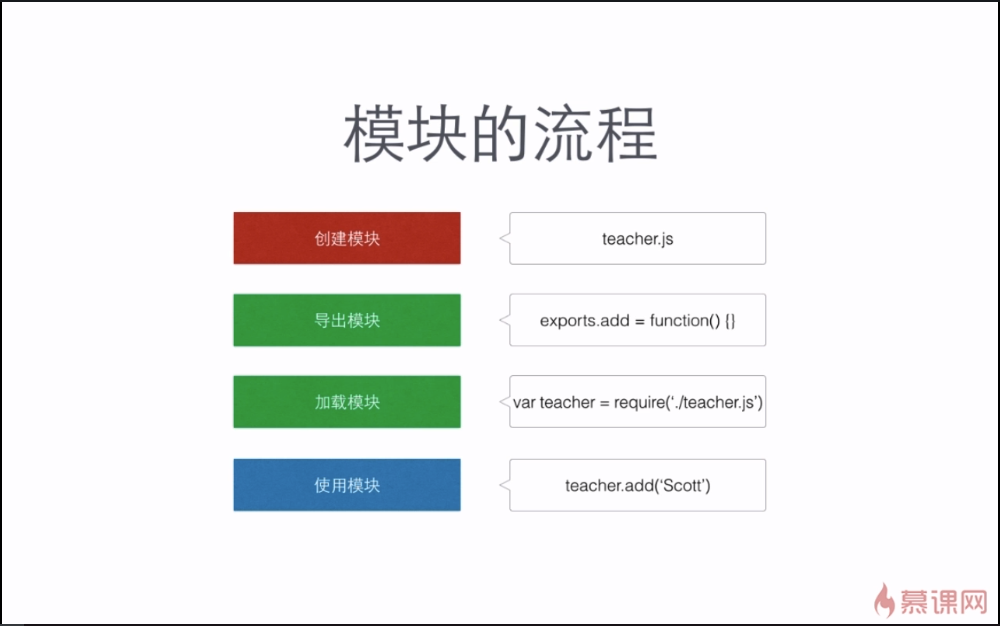
模块加载的顺序，按照其在代码中出现的顺序。



1. 在nodejs中文件和模块是一一对应的，如下是引用模块的方法：



1. 创建简单的模块：



1. 1
2. 1
3. 1

# 进击Node.js基础（二）

1. 1
2. 1
3. 1
4. 1
5. 1
6. 1

# node+mongodb 建站攻略（一期）

1. 1
2. 1
3. 1
4. 1
5. 1

# node建站攻略(二期)——网站升级

1. 1
2. 1
3. 1
4. 1
5. 1
6. 1