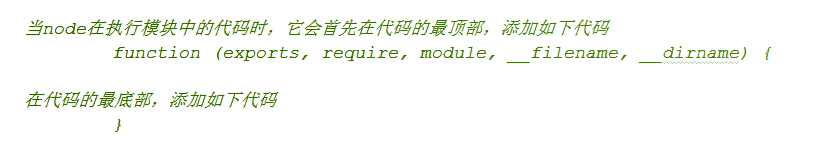
# Node的基础

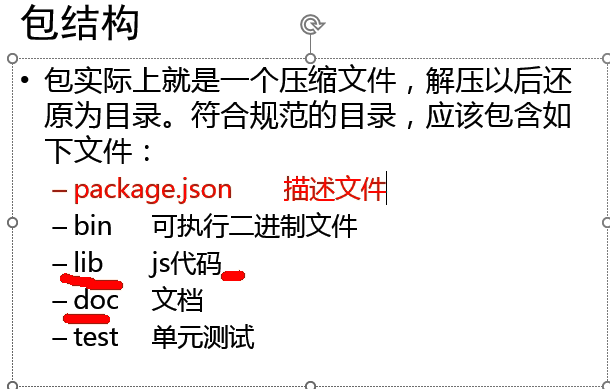
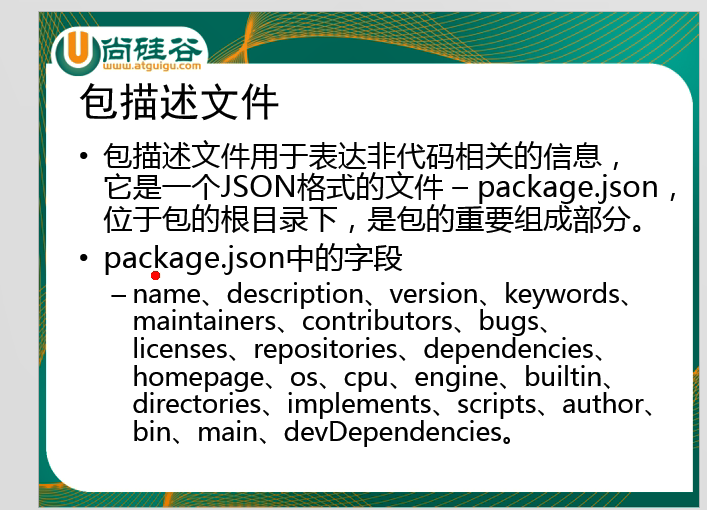
1. 在Node中有全局对象，作用和网页中的window类似
   1. 在全局中创建的变量都会作为global的 属性保存
   2. 在全局中创建的函数都会作为global的方法保存
2. Node除了主js文件是全局的，其他的模块js文件都是以函数运行的，即函数作用域
   1. 可以使用arguments.callee来验证模块js文件是函数对象
   2. Node在执行模块js代码时，都会添加：
      1. 
      2. 我们自己写的代码就是放在函数的中间
      3. 函数中的5个实参：
         1. Exports: 该对象属于module,用来将变量或者函数暴露到外部
         2. Require ： 函数，用来引入外部的模块
         3. Module ： 代表当前模块本身，
            1. exports就是module的属性，所以module.exports = exports
            2. 但是注意 不能用exports = { } 然后去暴露，会报错

因为本来exports = module.exports 的，所以能用exports.xxx,相当于往module.exports对象中添加属性

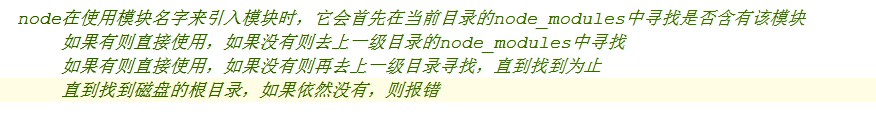
但是如果exports = {} 的话就把module.exports切断了，将exports弄成了自定义对象。跟module完全没关系了。

* + - 1. \_\_filename : 当前模块的完整路径
      2. \_\_dirname : 当前模块所在文件夹　的完整路径

# Package

1. CommomJS的规范包必须由包结构和包描述文件两个部分组成
2. 包本身上就是增强版的模块
3. 包结构：
   1. 
   2. 除了package.json 其他的都是可有可无的
   3. 
4. Json文件里面不能写注释。

# NPM

1. 全名：Node Package Manager
2. Npm的常用命令
   1. Npm -v 查看npm的版本
   2. Npm version 查看所有模块的版本
   3. Npm search 包名
      1. Npm search 包名 安装
   4. Npm install/i 包名
   5. Npm remove/r 包名
   6. Npm install 包名 –save 安装包，并且添加到依赖。
      1. 为了让下载你的项目的用户能通过package.json找到你的项目所依赖的网上的那些包
   7. Npm install （下载当前项目所依赖的包）
   8. Npm install 包名 -g 全局包装包（全局包装的包一般都是一些工具）
   9. 中国的npm镜像：npm install -g cnpm --registry=https://registry.npm.taobao.org
3. 通过npm下载的包都放到node\_modules文件夹中
   1. 需要使用包是，直接用require(“包名”)就性。
   2. 

# Buffer缓冲区

1. Buffer的结构和数组很像，操作的方法也和数组类似
2. 相当于能够存储二进制的数组
3. 使用buffer不需要引入模块，直接使用即可
4. 在buffer总存储的都是二进制数据，但是在数组内都是以16进制（unicode编码）的形式显示的
   1. 每个元素的范围是00 – ff
5. Buffer的长度是表示占用的内存的大小（字节）
   1. 如果存有汉字的话，一个汉字是3个字节
6. Buffer的所有构造函数都不推荐使用了。
7. Buffer的长度一旦设定就不能修改
8. Buffer的方法
   1. Buffer.alloc(10) 创建一个内存大小为10的buffer
   2. Buffer.from(string) 将string的所有字符的unicode装进buffer
   3. Buffer.allocUnsafe( 10 ) 创建一个内存大小为10的buffer,但是buffer中可能含有敏感数据
   4. Buffer.prototpye.tostring() 可以将缓冲区buffer中的数据转换去文本数据

# 同步的文件写入

1. 使用文件系统，需要先引入fs模块，fs是核心模块，直接引入不需要下载
2. Fs模块中的所有的操作都有同步，异步两种形式可供选择、
3. 同步文件写入
   1. Fs.openSync(path,flags,mode)
      1. Path:文件的路径
      2. Flags 打开文件要做的操作的类型
         1. “r” 只读
         2. “w” 读取
      3. Mode ：设置文件的操作权限，一般不传
      4. 返回文件的描述符作为结果。通过此描述来操作次文件
   2. Fs.writeSync(fd,string,position,encoding)
      1. Fd 写入文件的描述(openSync的返回值)
      2. String 写入内容
      3. Position 写入的位置 一般不传
      4. Encoding 写入的编码模式，一般不传，默认
   3. File.closeSync（fd）
      1. 关闭文件并保存

# 异步文件写入

1. Fs.open(path ,flags ,callback( err,fd ){ err用来判断是否出错，fd是文件的标识，用来操作。})
2. Fs.write( fd,string,callback( err){ err用来判断是否出错 } )
3. Fs.close(fd,callback(err){ })
4. 实战：将异步文件写入操作一遍

# 简单的文件写入

1. Fs.writeFile(file,data[,options] ,callback)
   1. File 需要操作的文件
   2. Data 需要写入的信息
   3. Options 是个对象，有下列属性 可以对写入进行一些设置
      1. Encoding
      2. Flags
      3. Mode
   4. Callback 当写入完成以后执行
2. Fs.writeFileSync(file,data[,options])
3. Flag有下列值：

# 流式的文件写入

1. 同步，异步，简单文件的写入都不适合大文件的写入，性能较差，容易导致内存溢出
2. 创建一个可写流
   1. Let ws = Fs.createWriteStream(path[,options])
      1. Path : 文件路径
      2. Options 配置的参数
   2. Ws.write(string) 只要流还在，就可以一直写，循环写
3. 可以通过监听流的open和close事件监听流的打开和关闭事件
   1. Ws.on(“open”,function(){}) 监听open的事件
   2. Ws.on(“close”,function(){}) 监听close的时间
   3. 也可以使用once来监听，只监听一次
4. 关闭写入流
   1. Ws.end()

# 简单文件读取

1. Fs.readFile(path[,options],callback(err,data){})
   1. Data是读取到的文件的buffer
   2. 可以继续在回调函数中对data进行操作
2. Fs.readFileSync(path[,options])

# 流式文件读取

1. 创建一个可读流
   1. Let rs = Fs.createReadStream(path)
2. 为可读流绑定一个data事件，data事件绑定完毕，他就会自动读取数据
   1. rs.on(“data” ,function(data){ }) 读取到的内容会传入回调函数中的data参数中
      1. 每次读取到一定大小都会调用一次回调函数
      2. 如果读取的文件很大，就会调用多次回调
3. 可读流也可以通过open,close进行监听
4. Rs.pipe(ws)
   1. 可以直接将可读流读到的所有数据传入到可写流中
   2. 

# 文件系统的其他操作

1. fs.existsSync(path) 检查一个文件是否存在 返回布尔
2. fs.stat(path，callback(err,stat){}) 获取文件的状态，返回一个对象，对象中保存了当前对象状态的相关信息
   1. stat.size 文件的大小
   2. stat.isFile 是否是文件
   3. stat.isDirectory 是否是文件夹
3. fs.unlink(path,callback(err)) 删除文件
4. fs.readdir(path[,options],callback(err,files){}) 读取一个目录的目录结构
   1. files 所有文件的数组
5. fs.trancate(path,len,callback) 截断文件
6. fs.mkdir(path[,options],callback(){}) 创建一个目录
7. fs.rmdir(path[,options],callback(){}) 删除一个目录
8. fs.renaem(oldPath,newPath,callback) 对文件进行重命名（相当于剪切）
9. fs.watchFile(filename[,options],listener(curr,prev){})
   1. filename 监视的文件的名字
   2. options 配置选项对象
      1. interval : 表示目标应该每隔多少豪秒被轮询，默认值为5007
      2. persistent ：表明当前文件正在被监视，默认是true
   3. listener 回调函数，当文件发生变化时，回调函数会执行
      1. curr prev都是stat的对象，一个是当前文件的状态，一个是修改后的文件状态