Technology Tarena 达内科技

WEB培优 Node.js

Node.js Server-Side

Unit02

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	全局对象
	10:30 ~ 11:20	
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	模块系统
	15:00 ~ 15:50	
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑


全局对象
全局对象
全局对象
全局对象
global.console
global.process
定时器
global.setInterval
global.clearInterval
global.setTimeout
global.clearTimeout
global.setImmediate
global.clearImmediate

process.nextTick

更多的全局函数

更多的全局对象

Technology Tarena 达内科技

全局对象

全局对象

Technology Tarena 达内科技

知识讲解

- window？
  - 嵌入在浏览器中的V8引擎，支持ES原生对象、BOM和DOM对象，全局对象为BOM中的window；
  - 声明全局作用域的变量和函数默认为window对象的成员——全局对象污染！
- global！
  - 独立的Node.js环境中，不支持BOM和DOM对象，全局对象称为global/GLOBAL；

对象称为global/GLOBAL；

– 声明的全局变量和函数在交互模式下是global对象的成员——全局对象污染；而在脚本模式下不是global对象的成员——避免了全局对象污染。

global.console

Technology Tarena 达内科技

- global.console 对象类似于Chrome中的console对象，用于向stdout和stderr中输出信息。

console对象并非ES标准对象，很多浏览器都有提供，Node.js也提供了自己的实现

知识讲解

```
var data = { id: 3, count: 36 };

console.log( `COUNT IS: %d`, data.count );    //标准输出
console.info( `COUNT IS: ${data.count}` );    //log别名
```

```
console.error( `COUNT IS: ${data.count}` );    //标准错误
console.warn( `COUNT IS: ${data.count}` );    //error别名

console.trace('Stack Trace');    //向stderr输出栈轨迹信息
```

global.console（续1）

Technology Tarena 达内科技

知识讲解

```
var data = { id: 3, count: 36, list:[ ] };

//向stdout输出指定对象的字符串表示
console.dir(data);

//为真的断言，错误信息不会输出
console.assert(data.count > 0, 'COUNT IS ZERO');
```

```
//为假的断言，抛出AssertError对象，
//输出错误信息，且终止脚本执行
console.assert(data.list.length > 0, 'NO DATA AVAILABLE')
```

global.console（续2）

Technology Tarena 达内科技

知识讲解

```
//使用 console 对象测量代码的执行耗时：

console.time( 'LONG-LOOP' );                    //开始计时
for( var i=0;  i<1000;  i++ ){
                    ;
}
```

```
console.timeEnd( 'LONG-LOOP' );                    //计时结束

//输出结果：          LONG-LOOP:  523ms
```

global.process

Technology Tarena 达内科技

- process：Object
  - 表示执行当前解释器运行所在的进程对象。
  - 可以使用该对象获取当前操作系统及运行时信息，并操作脚本所在执行进程。

知识讲解

当操作系统启动 Node.js 解释器执行一个脚本文件时，会将必需的文件从文件系统调入内存，分配必需的内存

空间，执行其中的代码 —— 此过程就创建了一个执行进程。全局对象 global.process 就是这个进程的代码表示。

global.process（续1）

Technology Tarena 达内科技

- process 对象中与操作系统相关的成员：

```
process.arch                    //获取CPU架构类型
process.platform                //获取操作系统类型
process.env                     //获取操作系统环境变量

//获取高精度的计时器(纳秒级)
```

知识讲解

```
//获取高精度的计时器(纳秒级)
var start = process.hrtime() setTimeout(()=>{
    //获取两次计时器时间差
    var diff = process.hrtime(start);
    console.log('REAL DRUATION: %d', diff[0]*1e9+diff[1]);
}, 1000);
```

global.process（续2）

Technology Tarena 达内科技

- process 对象中当前解释器和脚本相关的成员：

知识讲

```
process.cwd()           //获取当前所在工作目录
process.execPath        //获取解释器所在目录
process.execArgv        //获取解释器执行参数
process.argv            //获取解释器运行时的命令行参数
```

解

```
process.versions          //获取Node.js版本信息
process.uptime()          //获取Node.js解释器运行时间(s)
process.memoryUsage()     //获取内存信息

process.pid               //获取进程ID号
process.kill( pid )       //向指定进程ID号发送退出信号
```

定时器

Technology Tarena 达内科技

- 定时器任务：在间隔指定的时间后执行的任务。由于Node.js是单线程处理模型，所有到期要执行的定时器任务统一进入一个事件循环队列，由Node.js解释器依次调用执行。

知识

讲解

- Node.js 提供了四种形式的定时器：

(1)process.nextTick()：本次事件循环结束时立即执行的定时器；

(2)global.setImmediate()：下次事件循环立即执行的定时器；

(3)global.setTimeout()：一次性定时器；

(4)global.setInterval()：周期性定时器。

global.setInterval

Technology Tarena 达内科技

- setInterval(callback, delay)：function
  - 设置一个按指定周期不断执行的定时器。

知

  - 每隔delay毫秒重复调用回调callback。注意，取决于外部

识讲解

因素，如操作系统定的器粒度及系统负载，实际间隔可能会改变。

- 间隔值必须在1-2147483647的范围内（包含1和2147483647）。如果该值超出范围，则该值被当作1毫秒处理。一般来说，一个定时器不能超过24.8天。
- 返回一个代表该定时器的引用。

global.clearInterval

Technology Tarena 达内科技

- clearInterval(timer)：function
  - 停止一个之前通过setInterval()创建的定时器。回调不会再被执行。

识讲解

知识讲解

```
var count = 0;
var timer = setInterval( function(){
    console.log( count++ );
    if( count>=5 ){
        clearInterval( timer );
    }
}, 1000);
console.log('任务已安排');
```

global.setTimeout

Technology Tarena 达内科技

- setTimeout(callback, delay) : function
  - 设置一个一次性定时器。

知识讲解

- 在至少delay毫秒后调用回调callback。实际延迟取决于外部因素，如操作系统定时器粒度及系统负载。
- 超时值必须在1-2147483647的范围内（包含1和2147483647）。如果该值超出范围，则该值被当作1毫秒处理。一般来说，一个定时器不能超过24.8天。
- 返回一个代表该定时器的引用。

global.clearTimeout

Technology Tarena 达内科技

- clearTimeout(timer)：function
  - 停止一个之前通过setTimeout()创建的定时器。回调不会再被执行。

知识讲解

```
var count = 0;
var timer = setTimeout( function(){
    console.log( count++ );
    if( count<5 ){
        setTimeout( arguments.callee, 1000);
    } else {
        clearTimeout( timer );
    }
}, 1000);
console.log('任务已安排');
```

global.setImmediate

Technology Tarena 达内科技

- setImmediate(callback) : function
  - 设置一个尽可能立即执行的异步任务——下次事件循环开始时

始的。

知识讲解

- 语义上，相当于setTimeout(callback, 0)；但最终的执行时机，决定于系统定时器粒度及系统负载。
- 返回一个代表该定时器的引用。

global.clearImmediate

Technology Tarena 达内科技

- clearImmediate(timer) : function
  - 停止一个之前通过setImmediate( )创建的定时器。回调不

会再被执行。

知识讲解

```
var count = 0;
var timer = setImmediate( function(){
    console.log( count++ );
    if( count<5 ){
        setImmediate( arguments.callee );
    } else {
        clearImmediate( timer );
    }
} )
console.log('任务已安排');
```

process.nextTick

Technology Tarena 达内科技

- process.nextTick 定时器不同于 setTimeout(fn, 0) 或者 setImmediate(fn)，它指定的定时器任务在本次事件

者 setImmediate(fn)，它指定的定时器任务在本次事件
循环结束时立即执行，先于IO事件回调或其它定时器任务。

知识讲解

```
process.nextTick( () => {
        //先于IO事件回调和其它定时器回调
        console.log('nextTick任务...');
});

console.log('任务已安排');
```

更多的全局函数

Technology Tarena 达内科技

方法名	说明

知识讲解

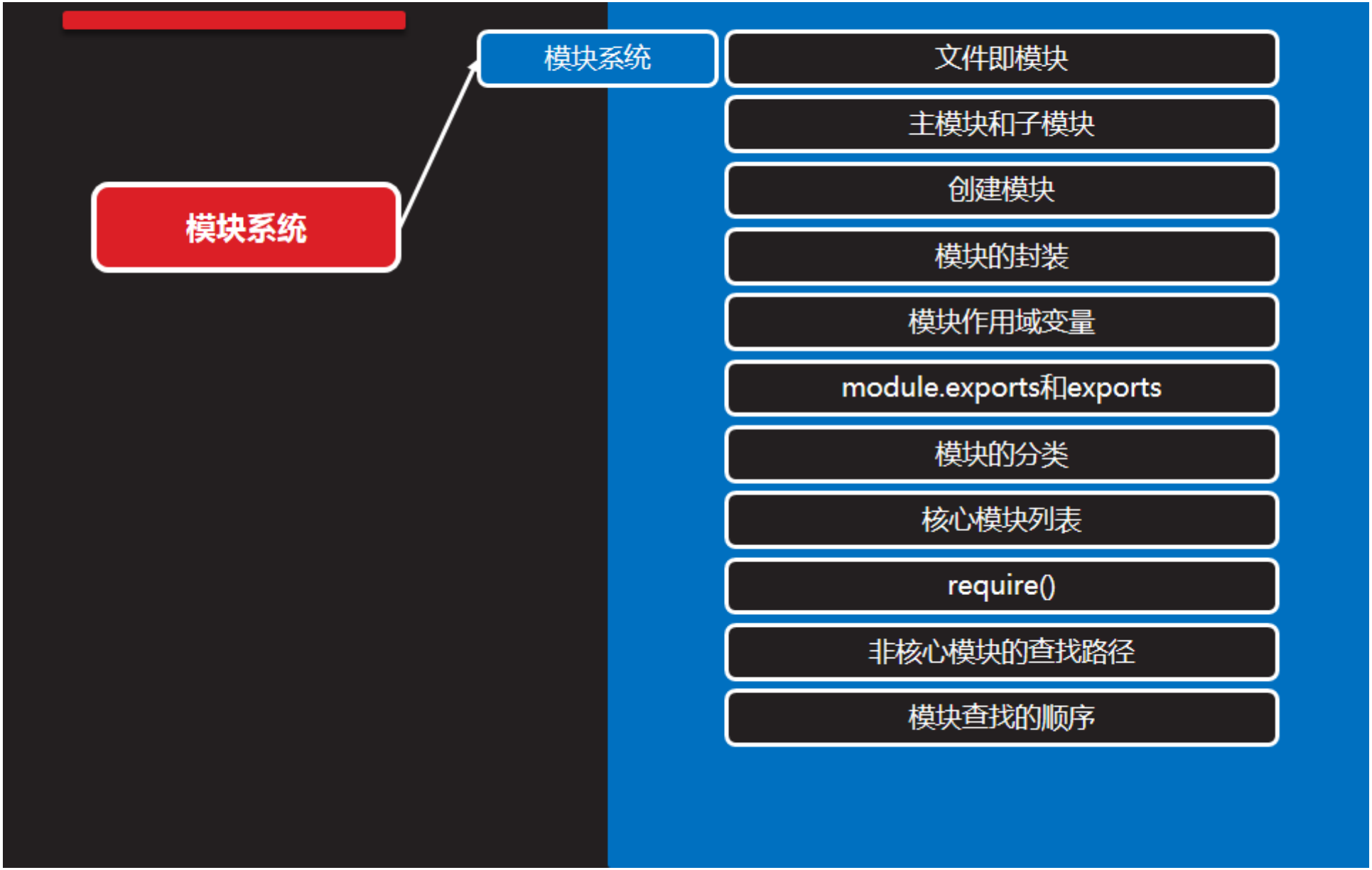
decodeURI( )	解码一个编码的 URI
decodeURIComponent( )	解码一个编码的 URI 组件
encodeURI( )	把字符串编码为 URI
encodeURIComponent( )	把字符串编码为 URI 组件
unescape( )	对由 escape() 编码的字符串进行解码
parseInt( )	解析一个字符串并返回一个整数
parseFloat( )	解析一个字符串并返回一个浮点数
isNaN( )	检查是否为NaN
isFinite( )	检查是否为有穷大的数字
eval( )	计算指定字符串，将其作为语句来执行

更多的全局对象

知识讲解

方法名	说明
JSON	JSON.parse( ) 和 JSON.stringify( )
Math	数学常量和数学函数
Array	构建数组对象
Boolean	构建布尔对象或布尔类型转换
Date	构建日期对象
Error (及其子类型)	构建错误及异常对象
Function	构建函数对象
Number	构建数字对象及数字类型转换
Object	构建对象
String	构建字符串对象及字符串类型转换
Buffer	构建内存缓冲区对象

模块系统



模块系统

文件即模块

知识讲解

- 类似于其它语言中的“包(package)”或“名称空间(namespace)”等概念，Node.js使用“模块(Module)”来规划不同的功能对象。
- Node.js 中每一个被加载的文件对应一个模块对象。
- 一个文件/模块被第一次加载后，会在内存中保存对应的缓存对象；对一个模块多次重复引入，会使用该缓存对象，从而避免了重复加载导致创建出多个完全相同的模块对象。

主模块和子模块

知识讲解

- 与 C 等其它语言类似，Node.js启动时运行的第一个模块称为“主模块”—— main module。
- 除主模块外的其它模块都称为“子模块”。
- 每个子模块都可以导出（exports）一些数据或方法供其它模块使用。
- 要使用其它模块的功能，当前模块需要引入（require）指定的模块。

```
//可以使用下述方法获取主模块对象
console.log( process.mainModule );
console.log( require.main );
//判断当前模块是否是“主模块”
console.log( module === process.mainModule );
```

创建模块

知识讲解

```
/**子模块：circle.js**/
const PI = 3.14;
//导出方法供其它模块使用
exports.size = function( r ){
        return PI * r * r ;
}
exports.perimeter = function( r ){
        return 2 * PI * r ;
}
```

```
/**主模块：app.js**/
const circle = require( './circle.js' );      //引用子模块
console.log(  circle.size( 3 ) );    //调用子模块提供的方法
console.log(  circle.perimeter( 3 ) );
```

模块的封装

Technology Tarena 达内科技

- Node.js中的每个.js文件都自成一个模块，有自己专属的成员属性和方法——“模块作用域”变量。模块文件中声明的变量和函数也都属于“模块作用域”。根本原因在于，Node.js在编译模块文件时会对其首尾进行如下的包装：

知识讲解

```
(function ( exports, require, module,
                                __filename, __dirname) {
        module.exports = { } ;
        exports = module.exports ;

        //模块文件中原有的全部内容

        return module.exports ;
 });
```

模块作用域变量

Technology Tarena 达内科技

- 声明在模块文件中的“全局变量”不再是真正的全局变量，即不是global对象的成员了；而是当前模块内部的局部变量。

知识讲解

成员名	说明
__dirname : String	当前模块文件所在的目录名
__filename : String	当前模块文件的文件名
module : Object	指向当前模块的引用
module.exports : Object	当前模块中待导出的供其它模块使用的对象
exports : Object	指向 module.exports 对象的引用
require : Function	引入其它模块，使用其它模块的 module.exports 对象

module.exports和exports

Technology Tarena 达内科技

知识讲解

- 一个模块可以导出一个供其它模块使用的对象，这个对象就是 module.exports 对象。起始时，它是一个 { } 对象。
- exports对象作为 module.exports 对象的引用，可以作为module.exports 的简写形式使用。
- 注意：其它模块可以引入的是 module.exports 对象，而不是 exports 对象！所以，下面两种情形的结果是不同的：

module.exports和exports（续1）

Technology Tarena 达内科技

知识讲解

```
exports.add = function(){
}
等同于
module.exports.add = function(){
}
```

```
exports = function(){        //无效
}
不同于
module.exports = function(){
}
```

模块的分类

Technology Tarena 达内科技

知识讲解

(1)核心模块

被编译进二进制执行文件，可以被解释器直接使用，加载速度最快；

(2)文件模块

没有后缀名的文件模块，被作为JavaScript文本加载；

.js 后缀的文件模块，被作为JavaScript文本加载；

.json 后缀的文件模块，被作为JSON字符串加载；

.node 后缀的文件模块，被作为C/C++二进制插件加载；

(3)目录模块

包含 package.json、index.js、index.json、index.node

包含 package.json、index.js、index.json、index.node 文件的目录；

核心模块列表

Technology Tarena 达内科技

知识讲解

模块名	说明
global	全局对象模块
console	控制台模块
util	提供常用函数的集合，用于弥补核心JS的功能过于精简的不足
events	实现了Node.js的事件驱动型的异步调用
fs	文件系统I/O操作模块
http	提供基于HTTP协议的请求和响应

require()

Technology Tarena 达内科技

- require( ) 函数用于引入另一个模块，其可用的参数有如下形式：

知识讲解

```
/** 核心模块名 **/
const http =  require( 'http' );
```

```
/** 相对或绝对路径开头的模块名 **/
const c1 =  require( './calc' );
const c2 =  require( '../calc' );
const c3 =  require( '/calc' );
```

可能是文件模块或目录模块

```
/** 不以路径名开头的非核心模块名 **/
```

```
/** 不以路径名开头的非核心模块名 **/
const mysql = require( 'mysql' );
```

非核心模块的查找路径

Technology Tarena 达内科技

- 对于每一个被加载的文件模块，创建这个模块对象的时候，module 对象便有一个paths 属性，用于指定查找不以“./”或“../”或“/”开头的模块名时所用到的路径。其值例如：

知识讲解

```
[ 'c:\\mynode\\project1\\lib\\node_modules',
  'c:\\mynode\\project1\\node_modules',
  'c:\\mynode\\node_modules',
  'c:\\node_modules' ]
```

模块查找的顺序

Technology Tarena 达内科技

（1）文件/目录模块的缓存

（2）原生模块缓存

（3）原生模块

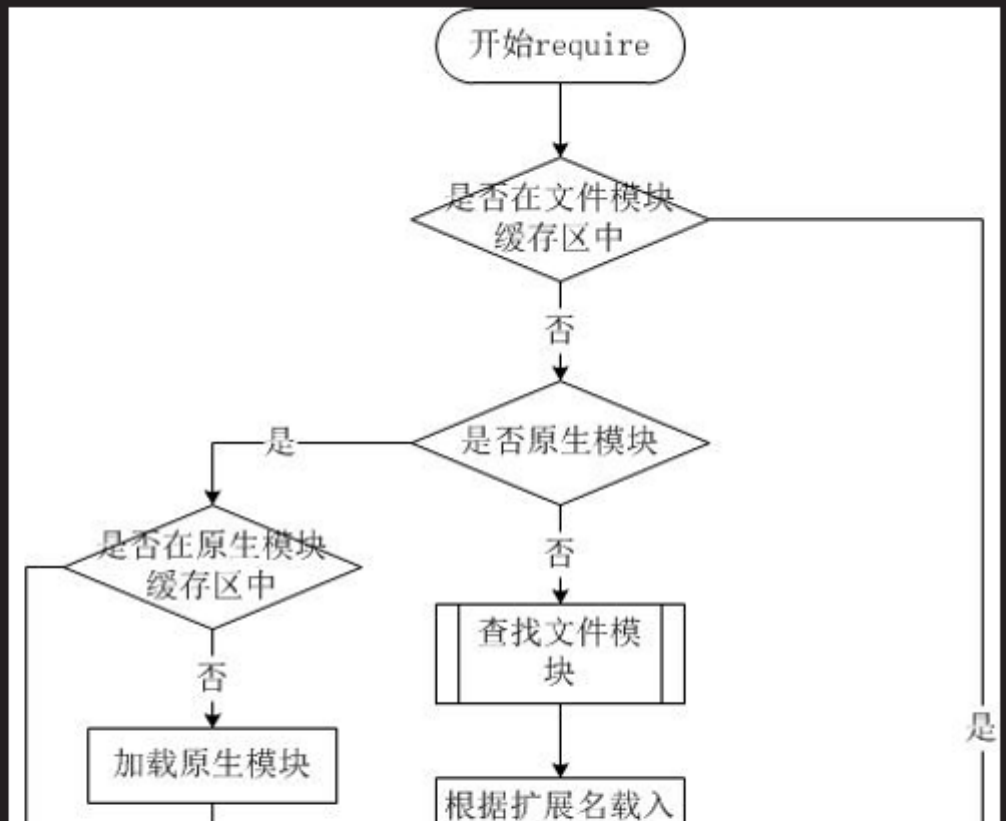
（4）文件/目录模块

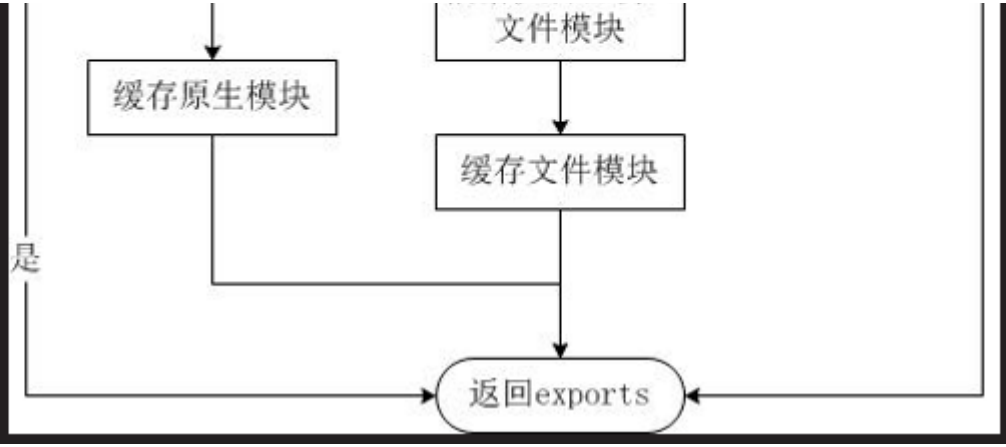
知识讲解

模块查找的顺序（续1）

Technology Tarena 达内科技

知识讲解





Technology Tarena 达内科技

总结和答疑