Technology Tarena 达内科技

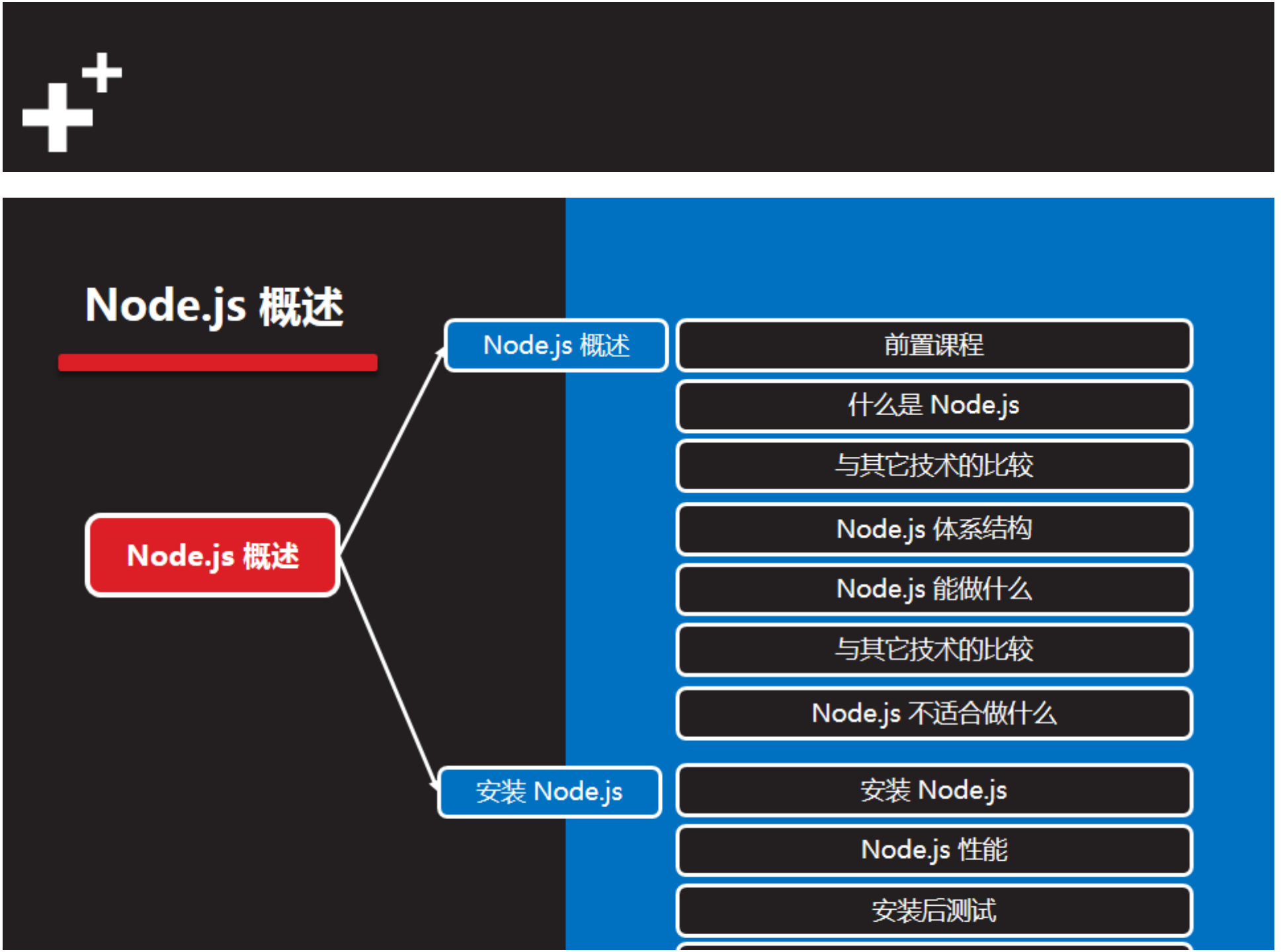
WEB培优 Node.js

Node.js Server-Side

Unit01

内容

上午	09:00 ~ 09:30	Node.js 概述
	09:30 ~ 10:20	
	10:30 ~ 11:20	
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	Node.js 基础语法
	15:00 ~ 15:50	函数和对象
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑


Node.js 概述
Node.js 概述
Node.js 概述
前置课程
什么是 Node.js
与其它技术的比较
Node.js 体系结构
Node.js 能做什么
与其它技术的比较
Node.js 不适合做什么
安装 Node.js
安装 Node.js
Node.js 性能
安装后测试

Node.js 两种运行模式

与 WebStorm 整合

安装后测试

Technology Tarena 达内科技

Node.js 概述

前置课程

Technology Tarena 达内科技

知识讲解

- (1) 有客户端 Javascript 使用基础，了解 JS 的基本语法以及常用对象，如 String、Math、Error、Object 等；
- (2) 本课程涉及到部分 HTML、CSS 相关知识；
- (3) 了解某一种服务器端编程语言，如 JSP、PHP 或 ASP.NET。

什么是 Node.js

Technology Tarena 达内科技

知识讲解

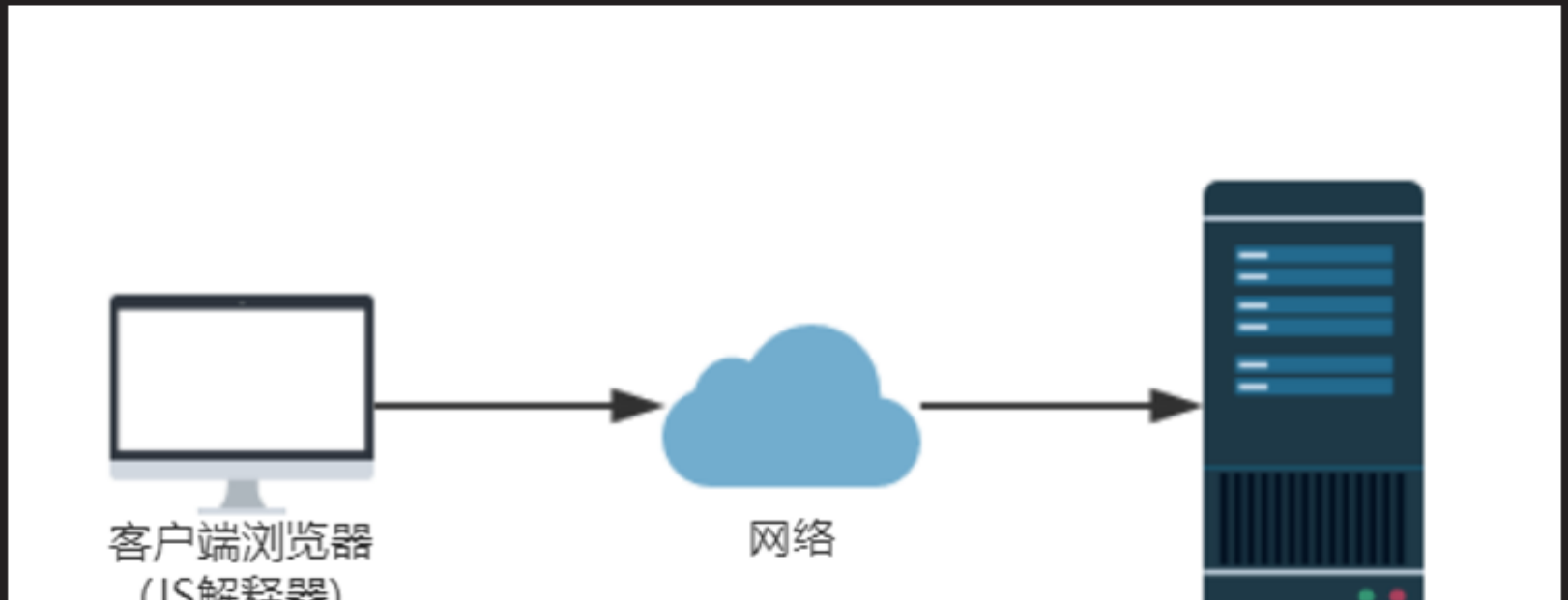
- 什么是Javascript？
  - 1995年由Netscape公司推出，后经ECMA统一标准的基于对象的脚本语言，由浏览器内置的JS解释器解释执行。JS对象由ECMAScript对象、DOM对象、BOM对象三部分组成。
- 什么是 Node.js？
  - 2009年由Ryan Dahl开发，现由Node.js Foundation维

护，基于Google V8引擎的JS运行时环境，其运行完全脱离浏览器，可以编写独立的服务器端程序。Node.js基于ES对象，提供了更多的符合CommonJS规范的对象。

什么是 Node.js（续1）

Technology Tarena 达内科技

知识讲解



WEB服务器
(Node.js解释器)

什么是 Node.js（续2）

Technology Tarena 达内科技

知识讲解

与其它技术的比较

Technology Tarena 达内科技

- 目前常用的服务器端技术有：
  - JavaEE
  - .Net
  - PHP
  - Node.js

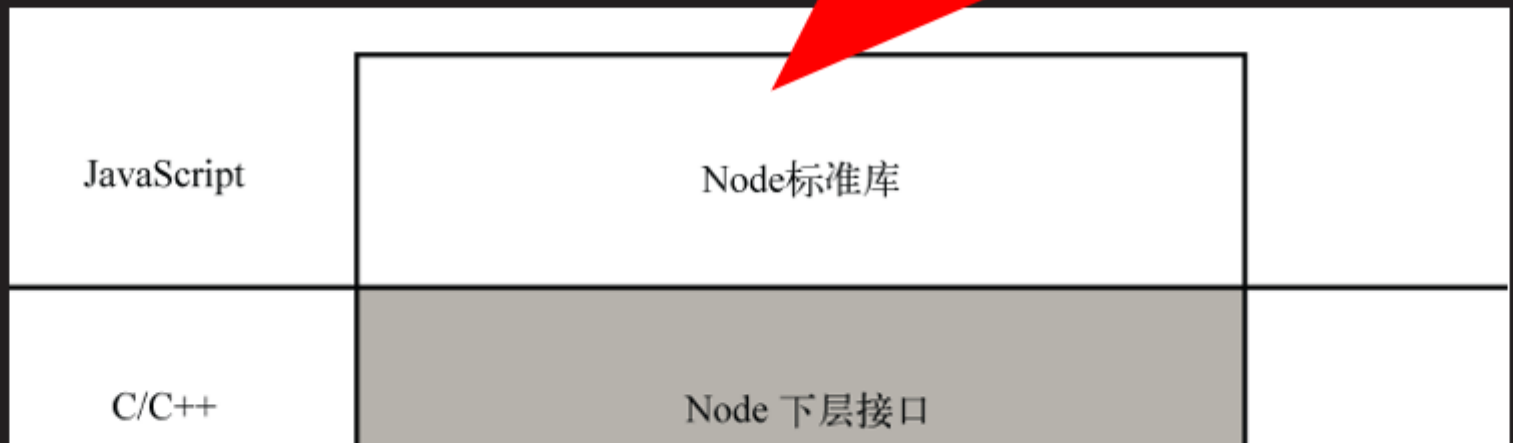
知识讲解

- Node.js是基于ECMAScript语言开发的服务器端语言，第一次使得前后端开发可以使用同一门语言来实现！
- 使用Node.js编写Web应用，无需借助于任何Web服务器——Node.js本身就可以编写Web服务器。
- Node.js应用结构简单，性能卓越！

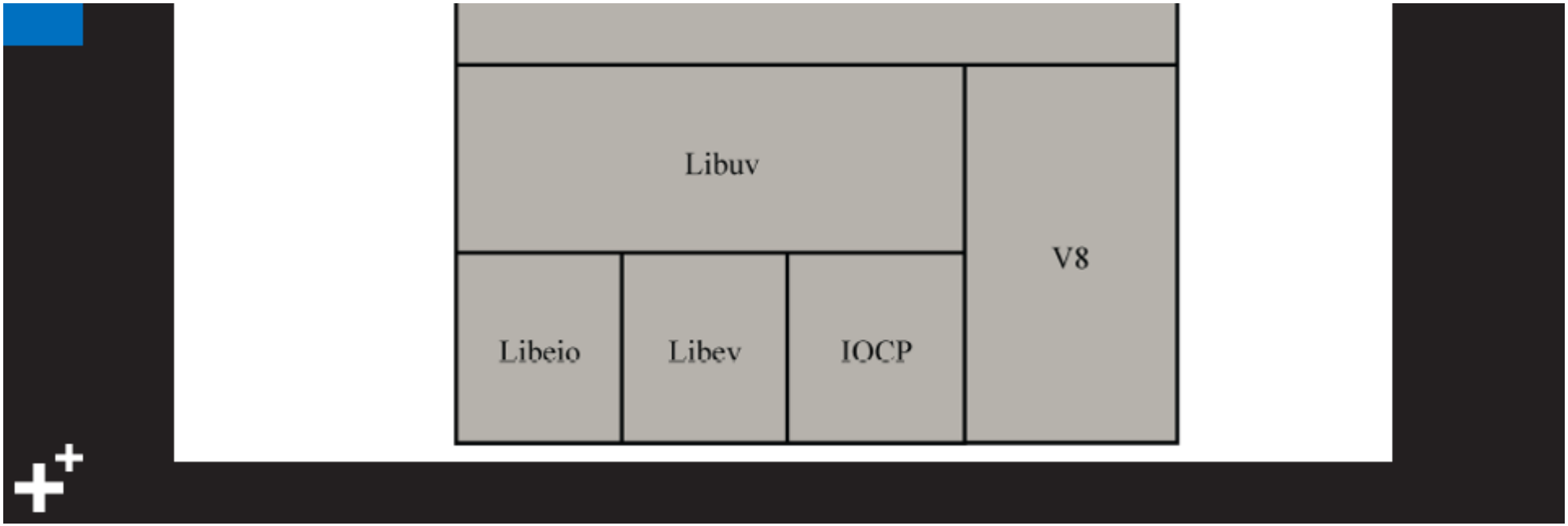
Node.js 体系结构

Technology Tarena 达内科技

https://nodejs.org/api/



知识讲解



Node.js 能做什么

Technology Tarena 达内科技

知识讲


解
集型应用
单元测试工具
WebSocket服务器
交互式终端程序


Node.js 不适合做什么
Technology
Tarena
达内科技
复杂加密和解密算法
知识

讲解

Node.js 特点

Technology Tarena 达内科技

- 简单，避免过度设计；
- 单线程逻辑处理；
- 非阻塞的异步I/O处理；

知

识讲解

- 事件驱动编程；
- 无锁机制，不会产生死锁；
- 支持数万个并发连接；

Node.js 性能

Technology Tarena 达内科技

- Node.js 与 PHP+Nginx 组合性能测试对比：
  （3000并发连接、持续30秒的压力下）
- **输出“hello world”响应：**
  - PHP 每秒响应请求数为3624，平均每个请求响应时间为

知识讲解

0.39秒；

– Node.js 每秒响应请求数为7677，平均每个请求响应时间为0.13秒；

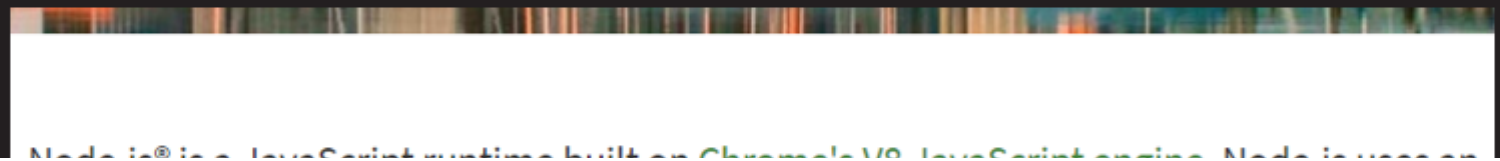
- **执行对MySQL的查询操作：**
  - PHP 每秒响应请求数为1293，平均每个请求响应时间为0.82秒；
  - Node.js 每秒响应请求数为2999，平均每个请求响应时间为0.33秒。

Technology Tarena 达内科技

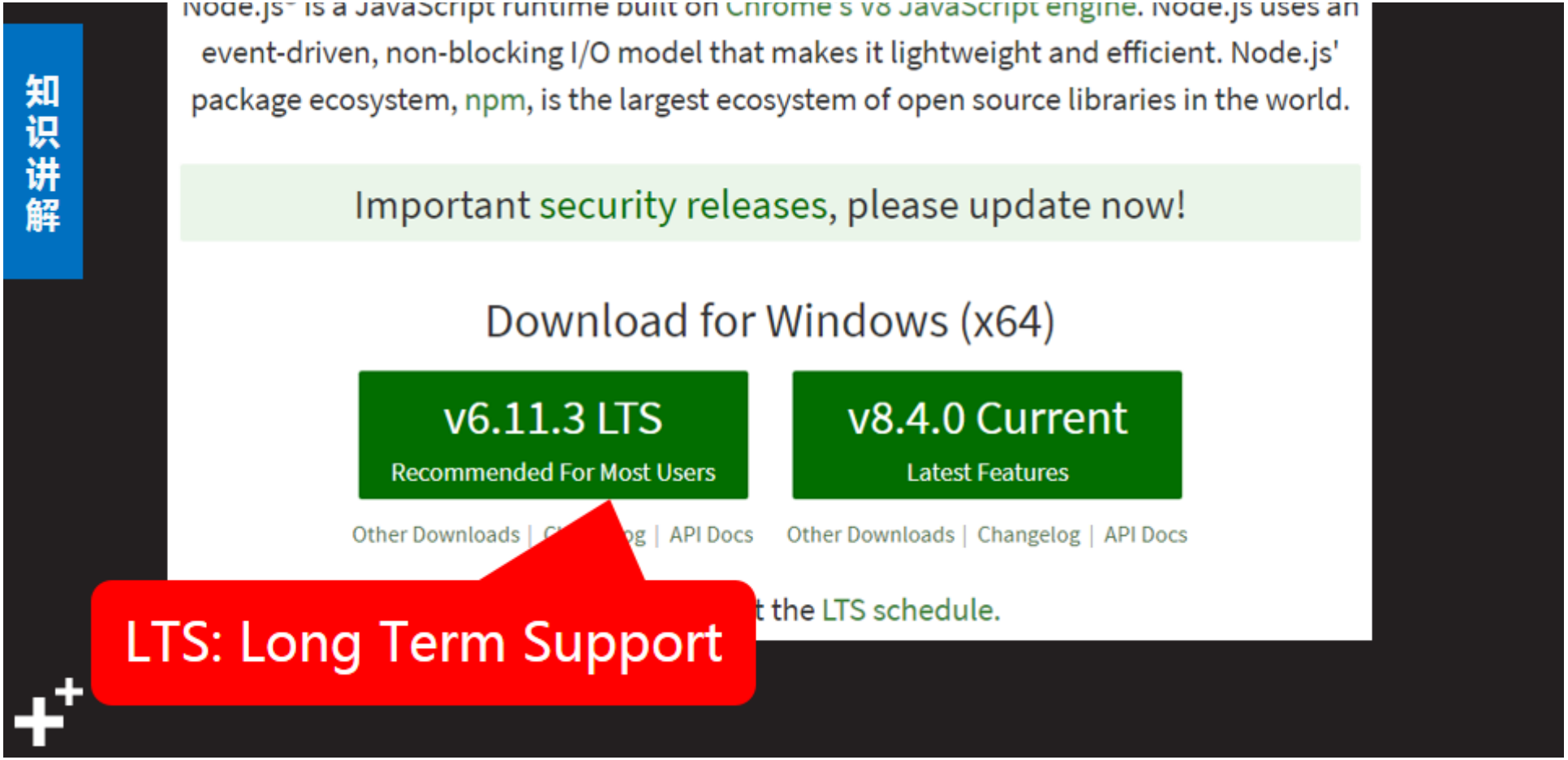
安装 Node.js

安装 Node.js

- 官方网站：https://nodejs.org/

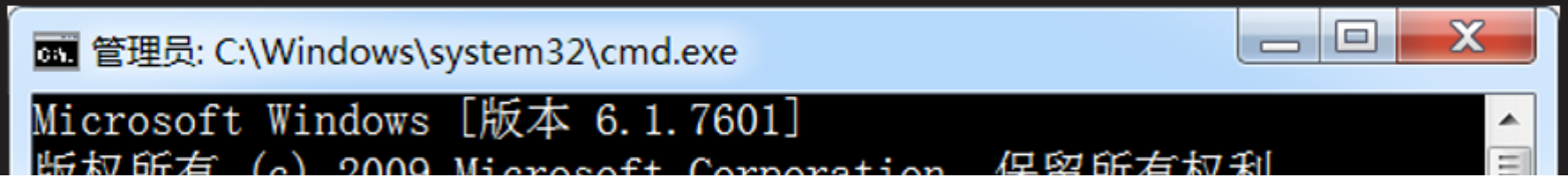


知识讲解

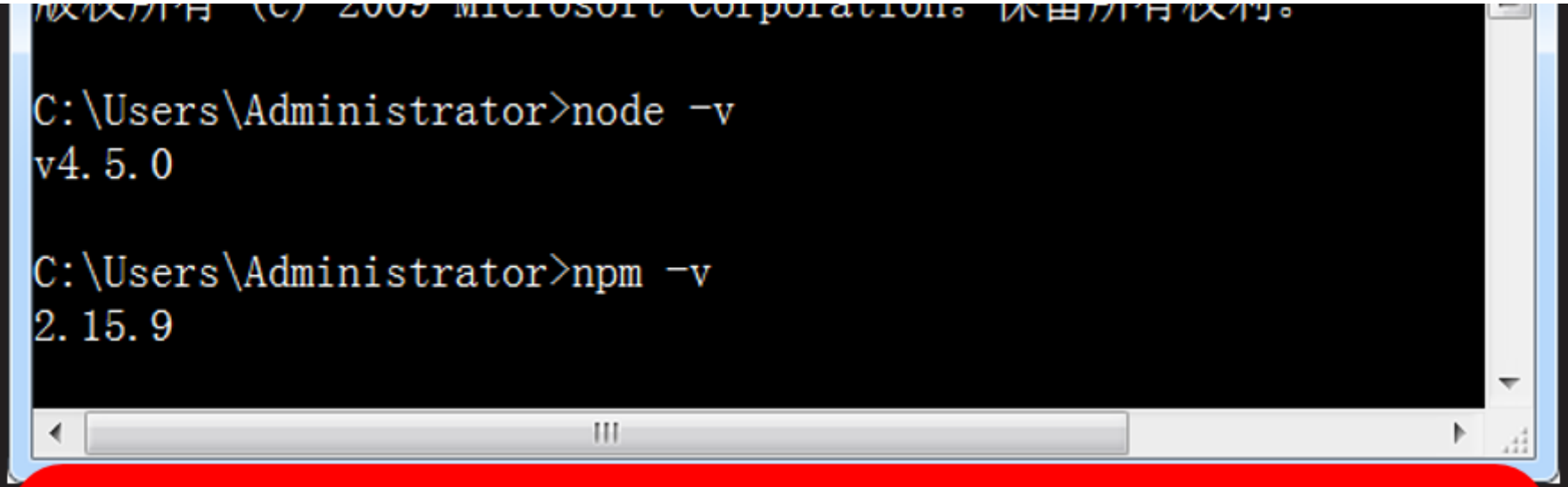


安装后测试

Technology Tarena 达内科技



知识讲解



Node.js 安装程序会自动配置Path环境变量，故无需手动修改环境变量。但卸载时不会自动清除安装时对Path的修改，必需手动删除，否则以后安装的Node.js仍然会先查找之前安装的版本。

Node.js 两种运行模式

Tarena Technology 达内科技

(1)交互模式

也称为 REPL 模式，Read-Evaluate-Print-Loop，读

取用户输入，执行运算，输出执行结果，继续下一次循环...

知识讲解

交互模式下，Node.js自带的模块无需使用 require() 引入

(2) 脚本模式

将所有语句编写在独立的脚本文件中，使用 node 命令一次性执行。

脚本模式下，除了全局对象及其相关成员外，所有其它模块中声明的对象和方法必须使用 require() 引入

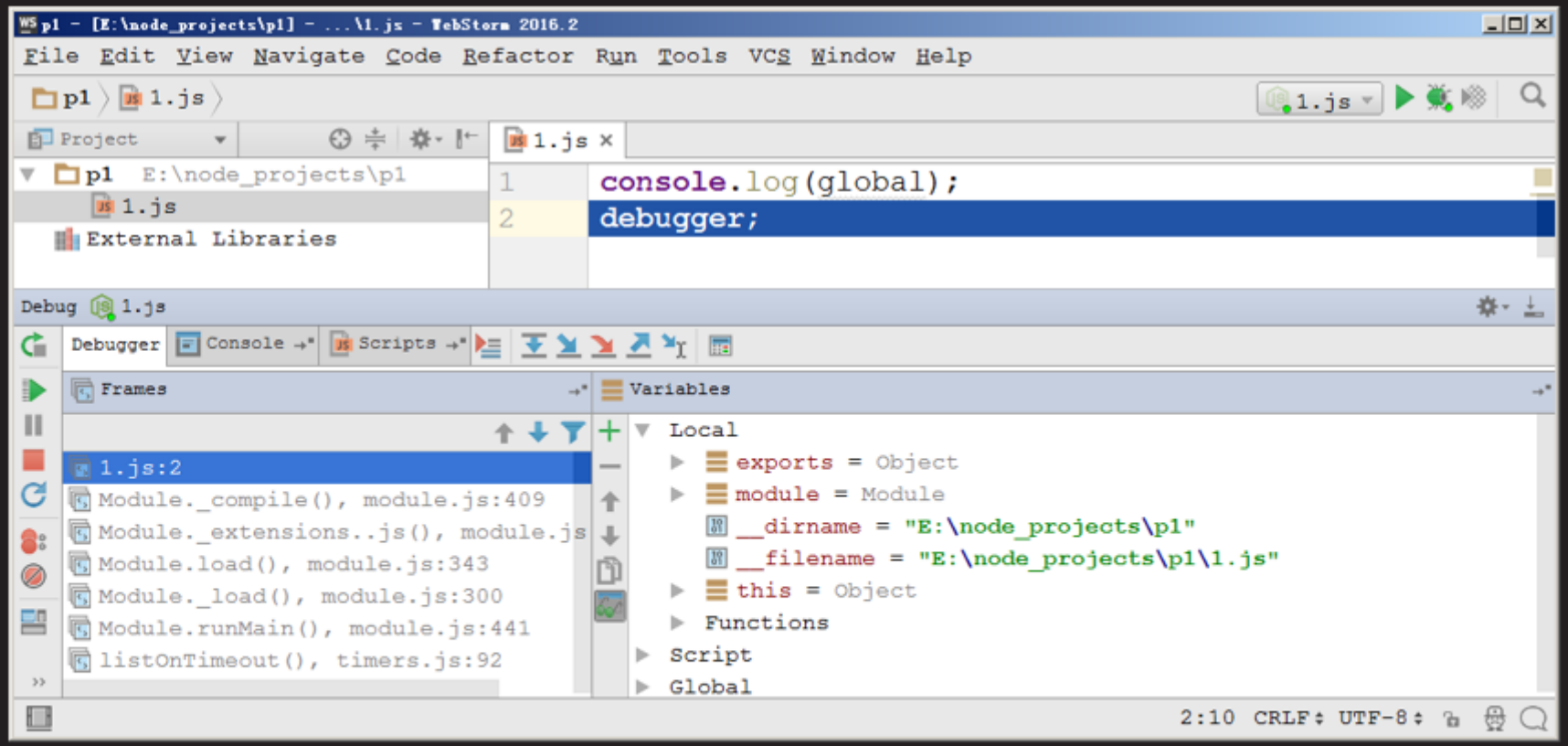
与 WebStorm 整合

Technology Tarena 达内科技

- WebStorm 是JetBrains公司旗下一款前端 IDE 工具，

与Intellij IDEA同源，具备强大的HTML/CSS/JS编辑提示、项目管理、运行及调试功能。

知识讲解



与 WebStorm 整合（续1）

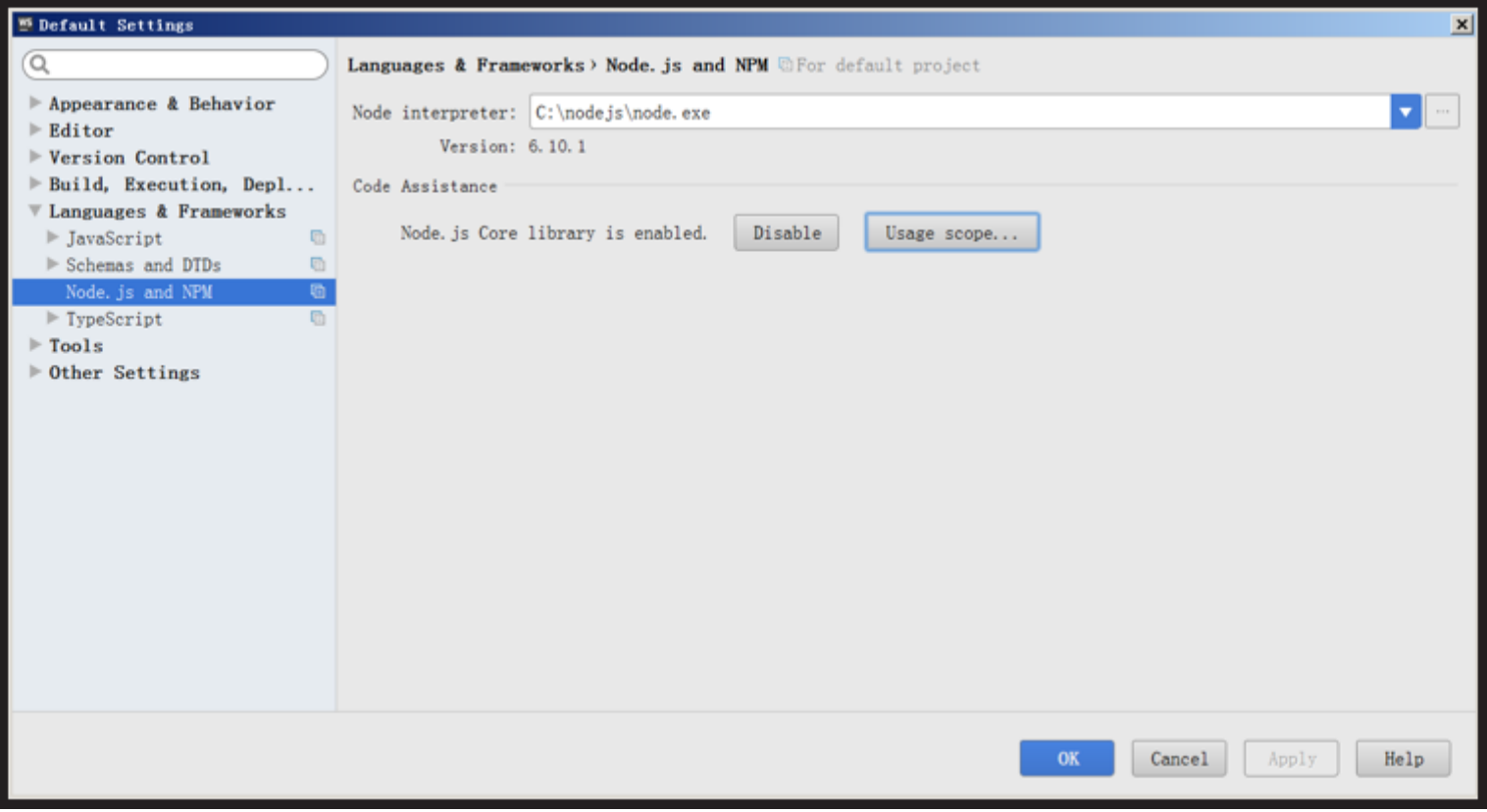
Technology Tarena 达内科技

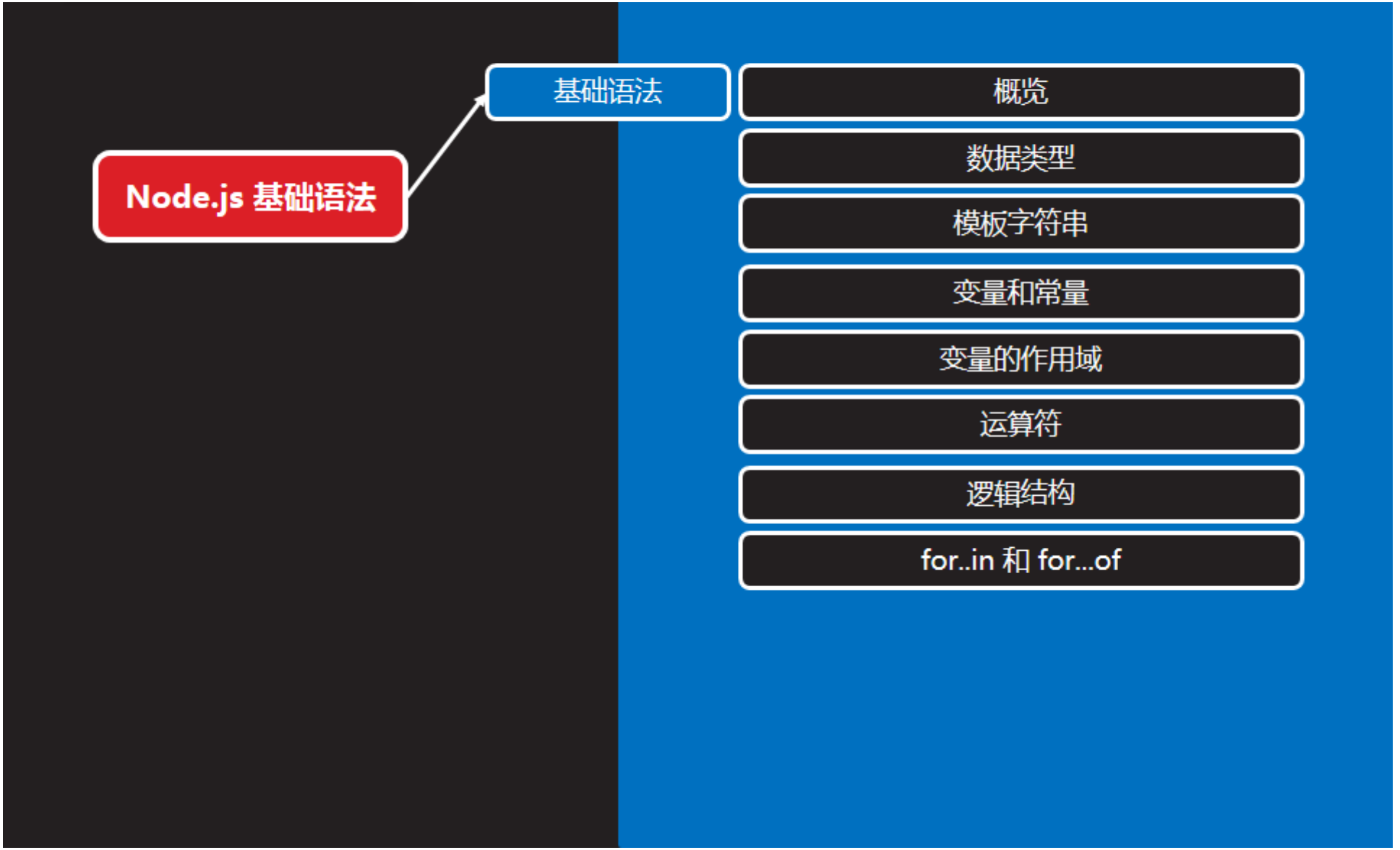
- WebStorm启动时会自动根据Path环境变量从前往后查

找到node解释器所在路径。

- 可以下载Node.js的源代码，并手工设置，这样WS就可以有更加丰富的Node.js代码提示功能及源代码查看功能。

知识讲解




Node.js 基础语法
基础语法
概览
数据类型
模板字符串
变量和常量
变量的作用域
运算符
逻辑结构
for..in 和 for...of

基础语法

知识讲解

- 数据类型
- 声明变量和常量
- 运算符
- 逻辑结构
- 函数和闭包
- 对象和继承

Node.js 从4.0开始，支持ES6的大多数新特性，例如：block scoping、template strings、for..of、arrow functions、collections、Promises、Symbols、class等。

数据类型

Technology Tarena

数据类型

知识讲解

(1) 原始类型（Primitive Type）：

String、Number、Boolean、Null、Undefined、Symbol；

原始类型数据直接赋字面值，或者调用对应的全局函数。

(2) 引用类型（Reference Type）：

包含ES 原生对象、Node.js对象、用户自定义对象三类；

引用类型数据一般需要使用 new 关键字进行创建。

模板字符串

知识讲解

- ES6标准中，可以使用“模板”方式定义字符串（Template String）：

```
var info = `
        用户名: 唐牧
        密码: 123456
`;
```

- 模板字符串中可以使用 ${ } 拼接变量，并执行运算：

```
var price = 3.5 , count= 3 ;
var info = `
        单价：${ price }
        数量：${ count }
        小计：${ price*count }
`;
```

变量和常量

Tarena Technology 达内科技

知识讲解

(1) 声明变量

```
let  i = 10 ;          //局部变量，ES6新特性
var j = 20 ;           //局部变量
k = 30;                //全局变量
```

注意：省略var的全局变量在严格模式下是非法的

(2) 声明常量 —— ES6新特性

```
const  fs = require('fs');
```

注意：严格模式下，修改常量的值是非法的

变量的作用域

Technology Tarena 达内科技

ES6中，变量的作用域分为三种：

(1)局部作用域：只能在当前函数内使用

(2)全局作用域：可以在任意函数内使用——是全局对象的成员

(3)块级作用域：只能在当前块内使用——ES6新特性

知识讲解

```
"use strict";

for( let i=0; i<10; i++ ){
    console.log(i);              //正常执行
}
console.log(i);                  //执行错误
```

运算符

知识讲解

- (1) 算术运算　　+　-　*　/　%　++　--
- (2) 比较运算　　>　<　>=　<=　==　===　!=　!==
- (3) 逻辑运算　　&&　||　!　&　|
- (4) 位运算　　&　|　~　^　<<　>>　>>>
- (5) 赋值运算　　+=　-=　*=　/=　%=
- (6) 三目运算　　?:
- (7) 特殊运算　　typeof　instanceof　void　,　.　[ ]　=>

逻辑结构

Tarena Technology 达内科技

知识讲解

(1)选择结构

- if...else....
- switch...case....

(2)循环结构

- for
- for...in...
- for...of... —— ES6新特性
- while
- do...while

for..in 和 for...of

Technology Tarena 达内科技

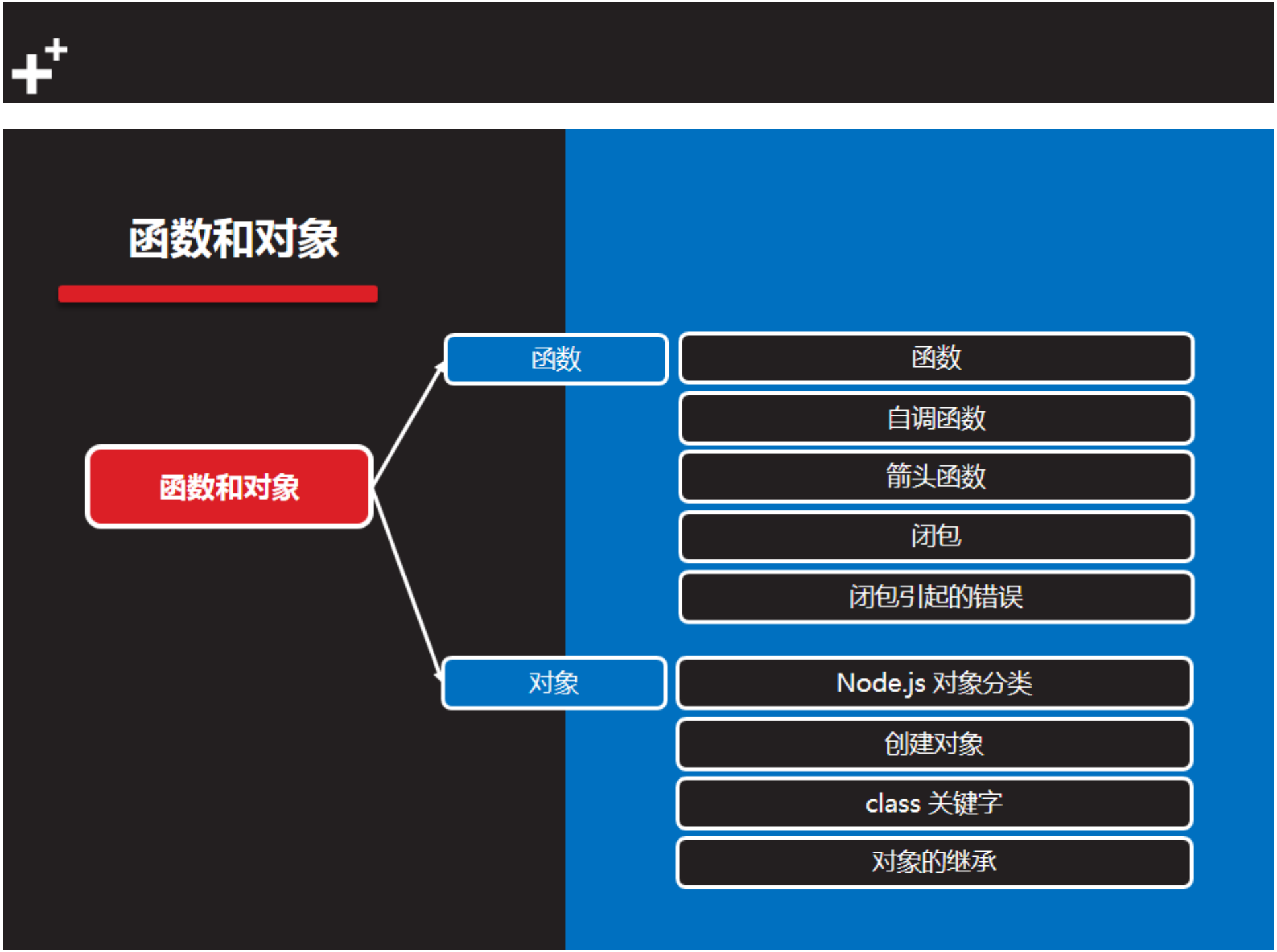
知识讲解

- for...in 循环可以遍历数组的下标或对象的成员名

```
var arr = [90, 80, 70];
for(var i in arr ){
    console.log( i );          //将输出 0  1  2
}
```

- ES6新增的 for...of 循环可以遍历数组的元素值

```
var arr = [90, 80, 70];
for(var v of arr ){
    console.log( v );          //将输出 90  80  70
}
```


函数和对象
函数和对象
函数
对象
函数
自调函数
箭头函数
闭包
闭包引起的错误
Node.js 对象分类
创建对象
class 关键字
对象的继承

函数

Technology Tarena 达内科技

- 函数：就是一系列语句的集合，为了实现某种可以重复使用的特定功能。

知识讲解

- (1)命名函数：指定了名称的函数对象

```
function add( num1, num2 ){
        return num1 + num2 ;
}
add( 10, 20 );
```

- (2) 匿名函数：未指定名称的函数对象

```
function( num ){
```

```
        arguments.callee( num-1 ); //递归调用匿名函数
}
```

自调函数

Technology Tarena 达内科技

- JS 中，全局变量会成为全局对象的成员，造成全局对象的污染；很多JS工具和框架在设计时，为了避免此问题，常常使用匿名函数的自调：

知识讲解

```
( function( g ){
    var ns = 'myNamespace';
} )( global );
```

```
+function( g ){
    var ns = 'myNamespace';
```

```
}( global );
```

箭头函数

Technology Tarena 达内科技

- ES6中，为了简化匿名回调函数的编写，引入了箭头函数

语法：

知识讲解

```
setInterval( ( )=>{
    console.log('timer...');
}, 1000)
```

```
fs.readFile('index.html', (err, data)=>{
    if(err) throw err;
    console.log(data);
```

```
} );
```

注意箭头函数中的“this指向”问题！

闭包

Technology Tarena 达内科技

- 闭包（Closure），是 ES 中所特有的现象。一个闭包，可以看做一种特殊的对象，包含若干函数对象，以及这些函数对象执行时必须依赖的执行上下文对象。

知识讲解

```
function outer( ){
        var num = 10;
        function inner(){
                console.log( num );
        }
        return inner;
```

```
}
var fn1 = outer( );          //创建了一个闭包对象
var fn2 = outer( );          //创建了一个闭包对象
```

闭包引起的错误

Technology Tarena 达内科技

- 无论是前端JavaScript还是后端Node.js，闭包使用不当，都可能引起看似奇怪的错误：

知识讲解

```
/**前端JS应用中闭包引起的错误**/

//点击一个按钮输出一个专有的数字
 var list = document.querySelectorAll('button');
 for(var i=0; i<list.length; i++){
          list[ i ].onclick = function(){
                    console.log( i );
```

```
        }
}
//点击每个按钮，会分别输出什么呢？
```

闭包引起的错误（续）

Technology Tarena 达内科技

- 使用Node.js，也可能产生类似前端应用中的闭包错误：

知识讲解

```
/**Node.js中闭包引起的错误**/

//目标：每隔1s，输出一个数字
 for( var i=0; i<3; i++ ){
        setTimeout( ()=>{
                console.log( i );
```

```
        }, 1000 );
}
//实际的运行结果呢？
```

Technology Tarena 达内科技

对象

Node.js 对象分类

Technology Tarena 达内科技

(1) ECMAScript 预定义对象

String、Boolean、Number、Date、Array、Math、RegExp、Function、Object、Error、EvalError、RangeError、ReferenceError、SyntaxError、TypeError、URIError 等等，以及更多的ES6新对象；

(2) Node.js 核心模块定义的对象

知识讲解

Buffer、WriteStream、Socket、ChildProcess 等等；

(3) 第三方模块定义的对象

(4) 用户自定义的对象

Node.js 中不支持 BOM和DOM对象，如window、document等！

创建对象

Technology Tarena 达内科技

- ECMAScript中提供了多种创建对象的语法：

(1)对象直接量方式

```
var e1 = { ename: '唐牧', work: function(){ } }
```

(2)构造函数方式

```
function Emp( ename ){
        this.ename = ename;
```

知识讲解

```
        this.work = function(){ }
}
var e2 = new Emp( '马丽' );
```

(3)原型继承方式

```
var parent = new Object( );
var child = Object.create( parent );
```

(4)class方式 —— ES6 新特性

class 关键字

Technology Tarena 达内科技

- ES6 借鉴了其它编程语言中声明对象的方式，正式启用了 class 关键字，有了“类”的概念。

知识讲解

- class：类，是一组相似对象的属性和行为的抽象集合。

解

- instance：实例，是从属于某个类的一个具体对象。

class 关键字（续1）

Technology Tarena 达内科技

知识

```
"use strict";
class Emp {                          //声明一个类
    constructor(ename){              //构造方法
        this.ename = ename;
    }
    work() {                         //实例方法
```

讲解

```
        console.log(`ENAME: ${this.ename}`);
    }
}

var e3 = new Emp( '岳寒' );        //创建类的实例
e3.work();
```

对象的继承

Technology Tarena 达内科技

- 继承，是面向对象编程的一个重要概念，使子对象可以继承父对象中声明的成员，从而极大的提高代码的复用性。

知识

- ES6 之前的继承都是通过对象的原型(prototype)来实现的：

识讲解

```
var graphic = { bgColor: 'red', borderColor: 'black' }
var rect = { width: 500,  height: 300 }

Object.setPrototypeOf( rect, graphic );     //原型继承
console.log(rect.width);
console.log(rect.height);
console.log(rect.bgColor);                  //继承来的成员
console.log(rect.borderColor);              //继承来的成员
```

对象的继承（续1）

Technology Tarena 达内科技

- ES6中的继承通过使用 class 配合 extends 关键字来实现。

```
/**首先声明父类**/
class Emp {                                 //声明父类
    constructor(ename){                     //父类构造方法
```

知识讲解

```
        this.ename = ename;
    }
    work( ) {
        return `ENAME: ${this.ename}`;
    }
}
```

对象的继承（续2）

Technology Tarena 达内科技

- ES6中的继承通过使用 class 配合 extends 关键字来实现。

```
/**再声明子类继承自父类**/
class Programmer extends Emp {
```

知识讲解

```
    constructor(ename,skills){
        super(ename);                    //调用父类构造方法
        this.skills = skills;
    }
    work(){                              //重写父类方法
        return super.work()+` SKILLS：${this.skills}`;
    }
}
var p1 = new Programmer('唐牧','JS');
console.log( p1.work() );
```

Technology Tarena 达内科技


总结和答疑