

浙江大学城市学院实验报告

课程名称 跨平台脚本开发技术

实验项目名称 实验九 模块与编程语言 Egg

学生姓名 吴成洋 专业班级 软件工程 1404 学号 31401417

实验成绩 指导老师（签名） 日期

注意：

- 务请保存好各自的源代码，以备后用。
- 请把作业保存为 pdf 上传到 BB 平台，请务必在截止日期前提交。

实验目的：

- 1、掌握 JS 中的模块定义与使用。
- 2、了解编程语言及其解释器的原理与实现

实验内容：

1. 阅读课件讲义并运行讲义示例程序
2. 教材 P145 习题 10.10.1，10.10.2，10.10.3
3. 教材 P156 习题 11.8.1，11.8.2，11.8.3

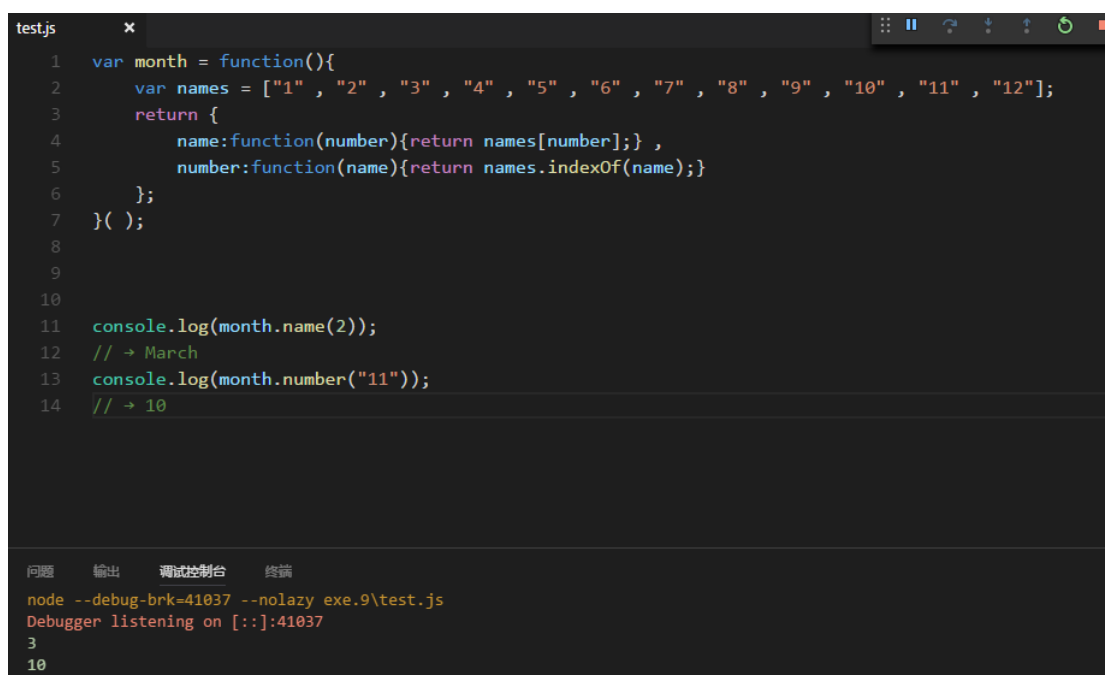
实验步骤：

1、

已阅读并运行课件 PPT

2、

2.1、



```
test.js x
1  var month = function(){
2      var names = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"];
3      return {
4          name:function(number){return names[number];} ,
5          number:function(name){return names.indexOf(name);}
6      };
7  }();
8
9
10
11 console.log(month.name(2));
12 // → March
13 console.log(month.number("11"));
14 // → 10

问题 输出 调试控制台 终端
node --debug-brk=41037 --nolazy exe.9\test.js
Debugger listening on [::]:41037
3
10
```

2.2、

分割模块，分为四个模块：

1. 用网格描述世界的点（1-6 函数）
2. 关于世界的行为和特定设定（7-9）
3. View 的增强（10-14）
4. 增加特色（15-18）

2.3、

require 命令的基本功能是，读入并执行一个 JavaScript 文件，然后返回该模块的 **exports** 对象。如果没有发现指定模块，会报错。

```
// example.js
```

```
var invisible = function () {  
  
    console.log("invisible");  
}
```

```
}
```

```
exports.message = "hi";
```

```
exports.say = function () {  
    console.log(message);  
}
```

运行下面的命令，可以输出 `exports` 对象。

```
var example = require('./example.js');  
  
example  
  
// {  
  
//   message: "hi",  
  
//   say: [Function]  
  
// }
```

如果模块输出的是一个函数，那就不能定义在 `exports` 对象上面，而要定义在 `module.exports` 变量上面。

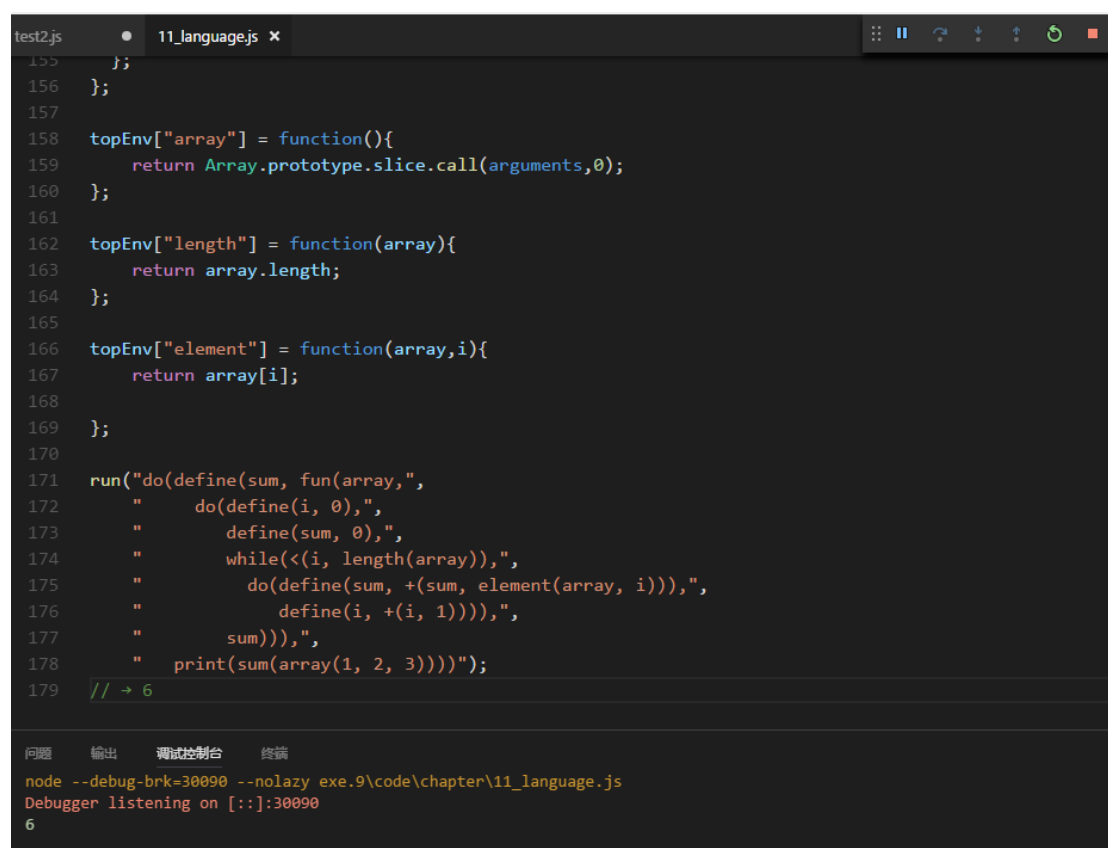
```
module.exports = function () {  
    console.log("hello world")  
}
```

```
require('./example2.js')()
```

上面代码中，`require` 命令调用自身，等于是执行 `module.exports`，因此会输出 `hello world`。

3、

3.1、



```
test2.js 11_language.js x
155 };
156 };
157
158 topEnv["array"] = function(){
159     return Array.prototype.slice.call(arguments,0);
160 };
161
162 topEnv["length"] = function(array){
163     return array.length;
164 };
165
166 topEnv["element"] = function(array,i){
167     return array[i];
168 };
169
170
171 run("do(define(sum, fun(array,",
172     "    do(define(i, 0),",
173     "        define(sum, 0),",
174     "        while(<(i, length(array)),",
175     "            do(define(sum, +(sum, element(array, i))),",
176     "                define(i, +(i, 1))),",
177     "        sum))),",
178     "    print(sum(array(1, 2, 3))))");
179 // → 6

问题 输出 调试控制台 终端
node --debug-brk=30090 --no-lazy exe.9\code\chapter\11_language.js
Debugger listening on [::]:30090
6
```

代码如下

```
topEnv["array"] = function(){
    return Array.prototype.slice.call(arguments,0);
};

topEnv["length"] = function(array){
    return array.length;
};
```

```
topEnv["element"] = function(array,i){
    return array[i];
};

run("do(define(sum, fun(array, ",
    "    do(define(i, 0),",
    "        define(sum, 0),",
    "        while(<(i, length(array)),",
    "            do(define(sum, +(sum, element(array, i))),",
    "                define(i, +(i, 1)))),",
    "            sum))),",
    "    print(sum(array(1, 2, 3))))");
// → 6
```

3.2、

由 fun 形式创建的函数会创建自己的局部环境,并将参数变量添加到其中,接着使用该环境对象执行函数体,并返回结果。

3.3、

```
21
22 function skipSpace(string) {
23     var skippable = string.match(/^(\s|#.*)*/);
24     return string.slice(skippable[0].length);
25 }
26
27 console.log(parse("# hello\nx"));
28 // → {type: "word", name: "x"}
29
30 console.log(parse("a # one\n  # two\n()"));
31 // → {type: "apply",
32 //     operator: {type: "word", name: "x"},
33 //     args: []}
34
```

问题 输出 调试控制台 终端

```
node --debug-brk=39788 --nolazy exe.9\code\chapter\11_language.js
Debugger listening on [::]:39788
[ type: 'word', name: 'x' ]
[ type: 'apply',
  operator: { type: 'word', name: 'a' },
  args: [] ]
```