



JavaScript

—— 作用域链 & 闭包 ——

自我介绍



王立 程序员
Nick Wang

百变空间CTO\联合创始人\董事、连续创业者
曾经在微软、阿里工作十年以上

18项发明专利

多家上市公司、基金顾问



Agenda

- ✓ 作用域链
 - ✓ 作用域
 - ✓ 执行环境和作用域链
- ✓ 闭包
 - ✓ 闭包原理
 - ✓ 应用
- ✓ 立即执行函数
- ✓ 动手实验 X 4
- ✓ 知识点 X 3

作用域

动手实验 9-1

- ✓ 函数作用域 [[scope]]
 - ✓ 外部对内部可见；
 - ✓ 内部对外部不可见；
 - ✓ 内部优先；
- ✓ JS中只有函数级别的作用域，没有块级别的作用域；换句话说，只有在进入或者退出函数的时候，作用域会发生变化。

执行环境和作用域链

知识点 9-1

- ✓ 执行环境 (execution context)，定义了执行期间可以访问的变量和函数。
 - ✓ 全局执行环境
 - ✓ Global Object (window)
 - ✓ 从见到JS代码开始创建
 - ✓ 到网页关闭时销毁
 - ✓ 函数执行环境
 - ✓ Activation Object
 - ✓ 从函数调用开始创建
 - ✓ 到函数调用结束时销毁

时间周期

执行环境（EC）和作用域链

知识点 9-1

- ✓ 作用域[[scope]]，每个函数都有。
- ✓ 作用域是私有属性，只能由JS引擎访问
- ✓ 作用域链，是AO和GO构成的链
- ✓ 所谓执行环境，就是根据作用域链依次查找变量和函数：
 - ✓ 找到即停；
 - ✓ 全部找完无果，报错。
- ✓ 作用域链每个函数都有

生成作用域链

知识点 9-2

- ✓ 每个函数在定义（函数声明、函数表达式）时会拷贝其父亲函数的作用域链；
EC
- ✓ 在函数被调用时，生成AO然后将AO压入作用域链的栈顶。

生成作用域链

知识点 9-2

✓ 1. 全局-预编译阶段

```
var g = 'g';  
function fa(){  
  var a = 'a';  
  function fb(){  
    var b = 'b';  
  }  
  fb();  
}  
fa();
```

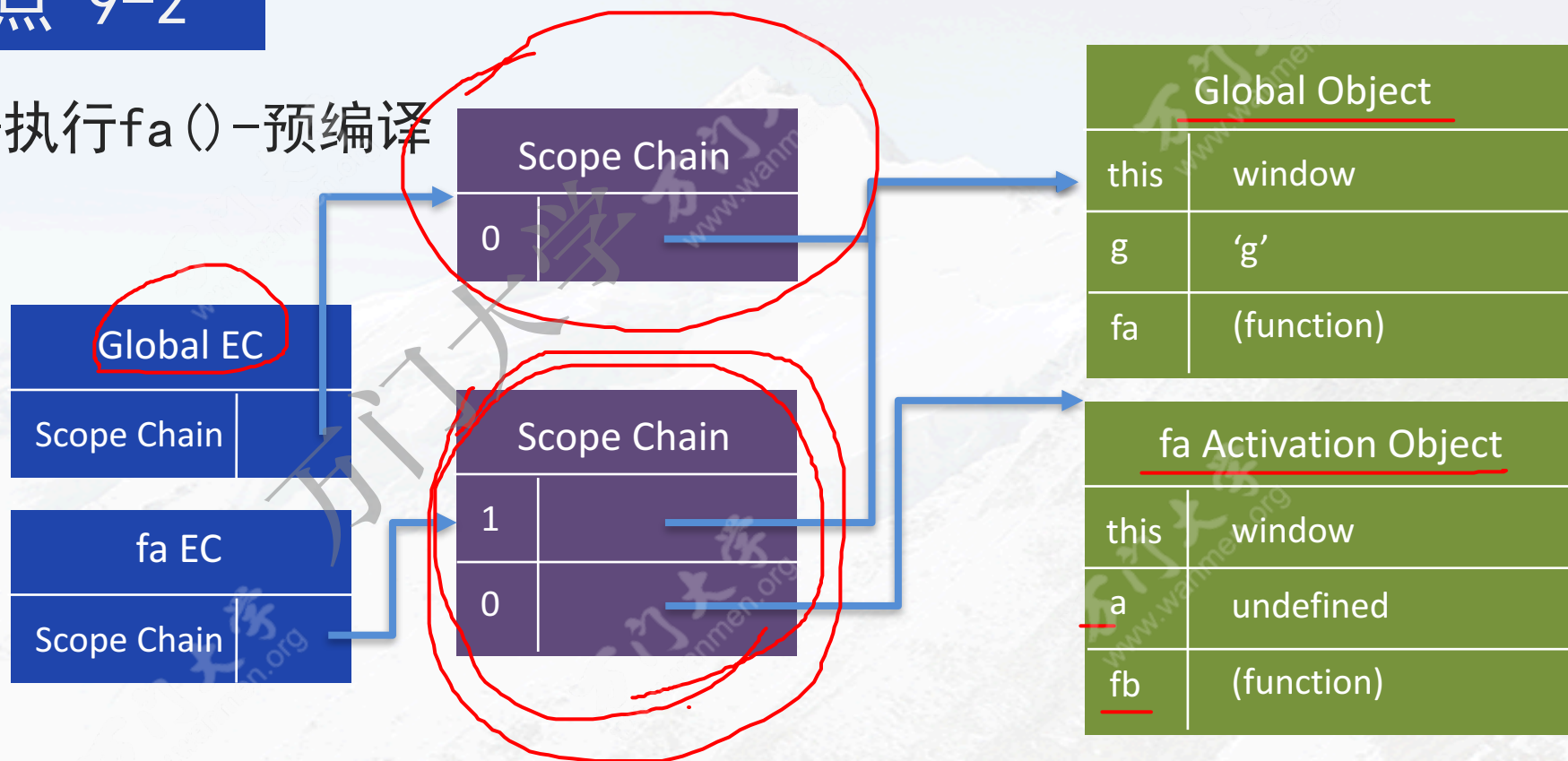


生成作用域链

知识点 9-2

✓ 2. 全局-执行fa()-预编译

```
var g = 'g';  
function fa(){  
  → var a = 'a';  
  → function fb(){  
    → var b = 'b';  
  }  
  → fb();  
}   
→ fa();
```

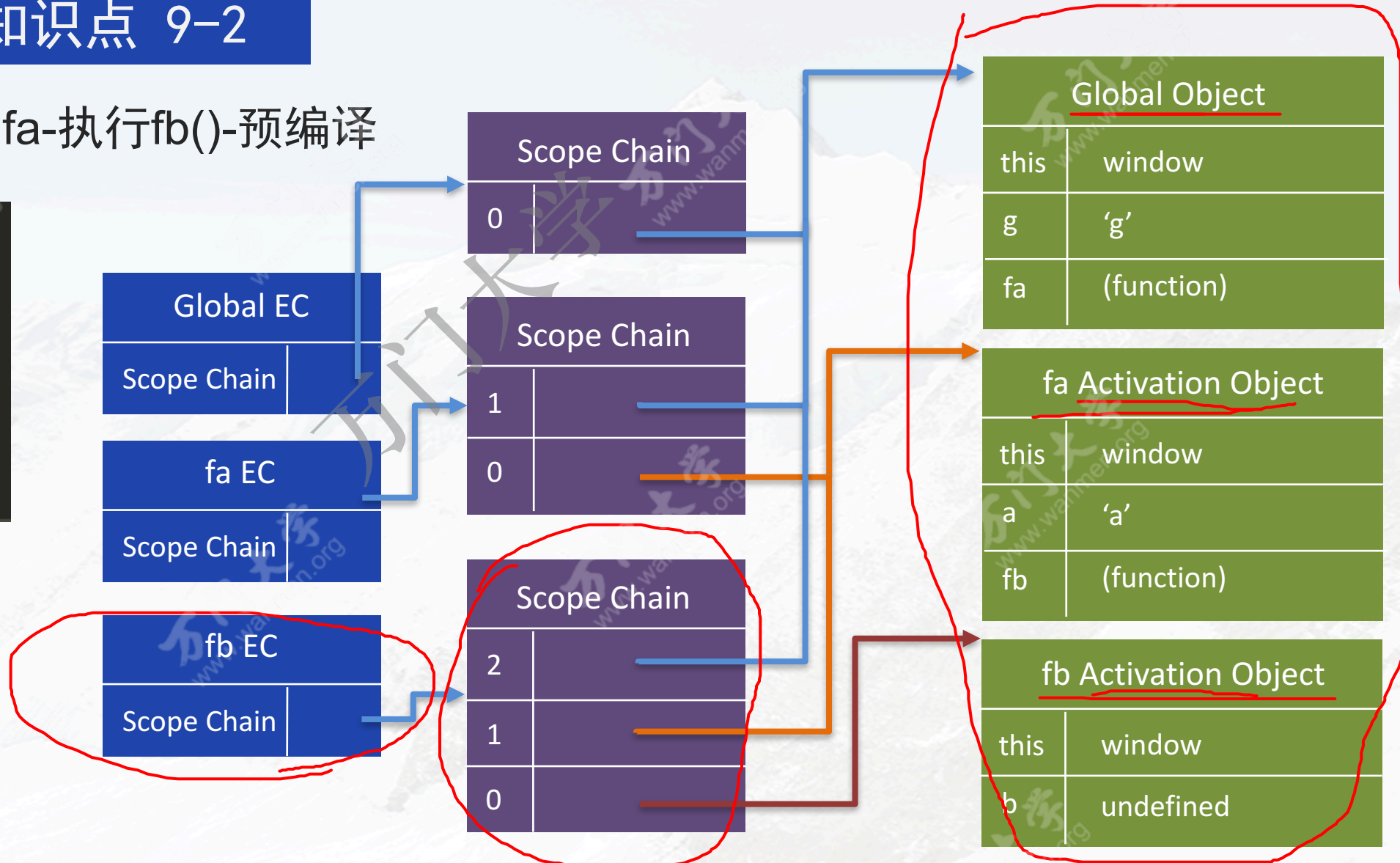


生成作用域链

知识点 9-2

✓ 3. fa-执行fb()-预编译

```
var g = 'g';
function fa(){
  var a = 'a';
  function fb(){
    var b = 'b';
  }
  fb();
}
fa();
```

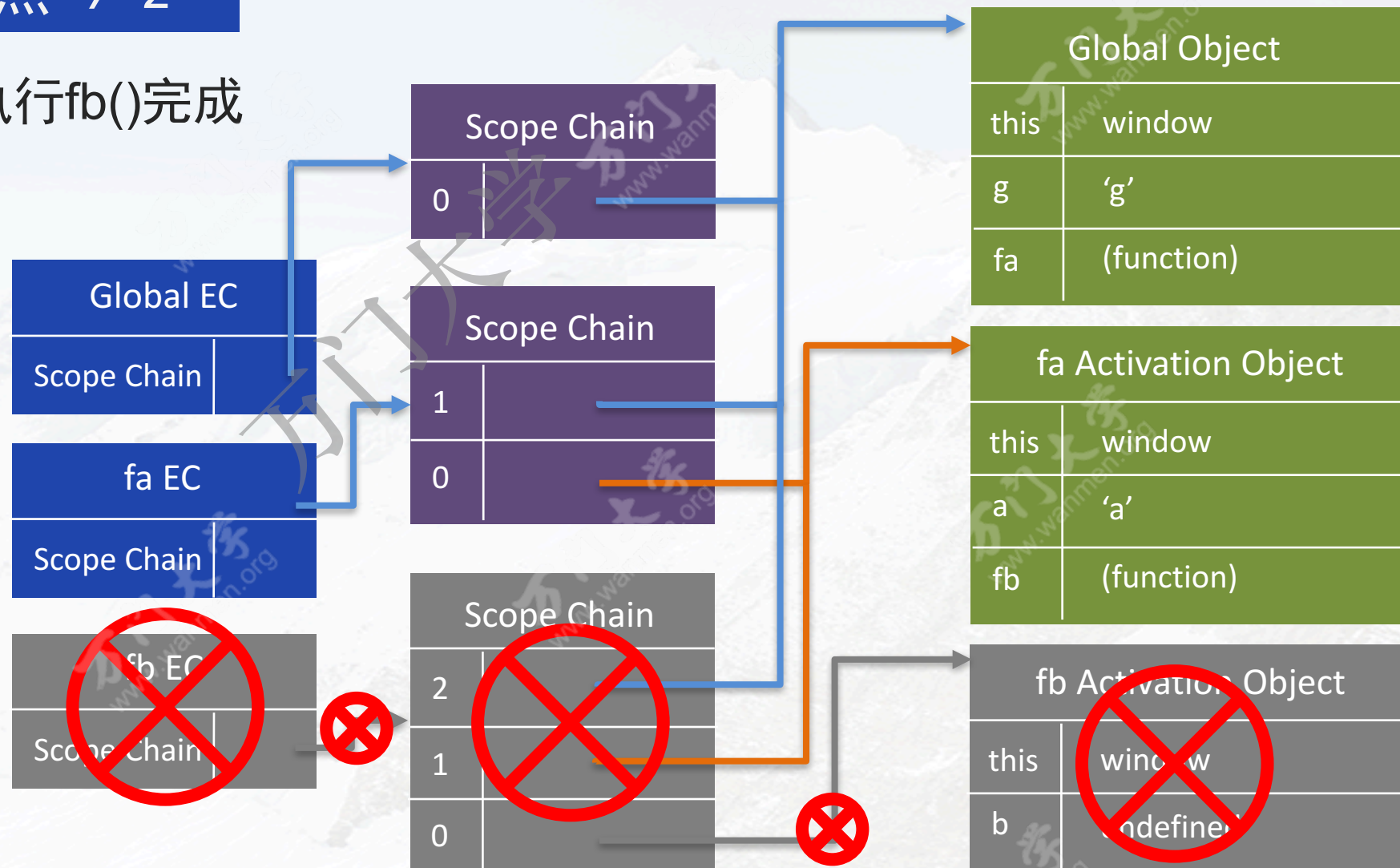


生成作用域链

知识点 9-2

4. fb-执行fb()完成

```
var g = 'g';
function fa(){
  var a = 'a';
  function fb(){
    var b = 'b';
  }
  fb();
}
fa();
```

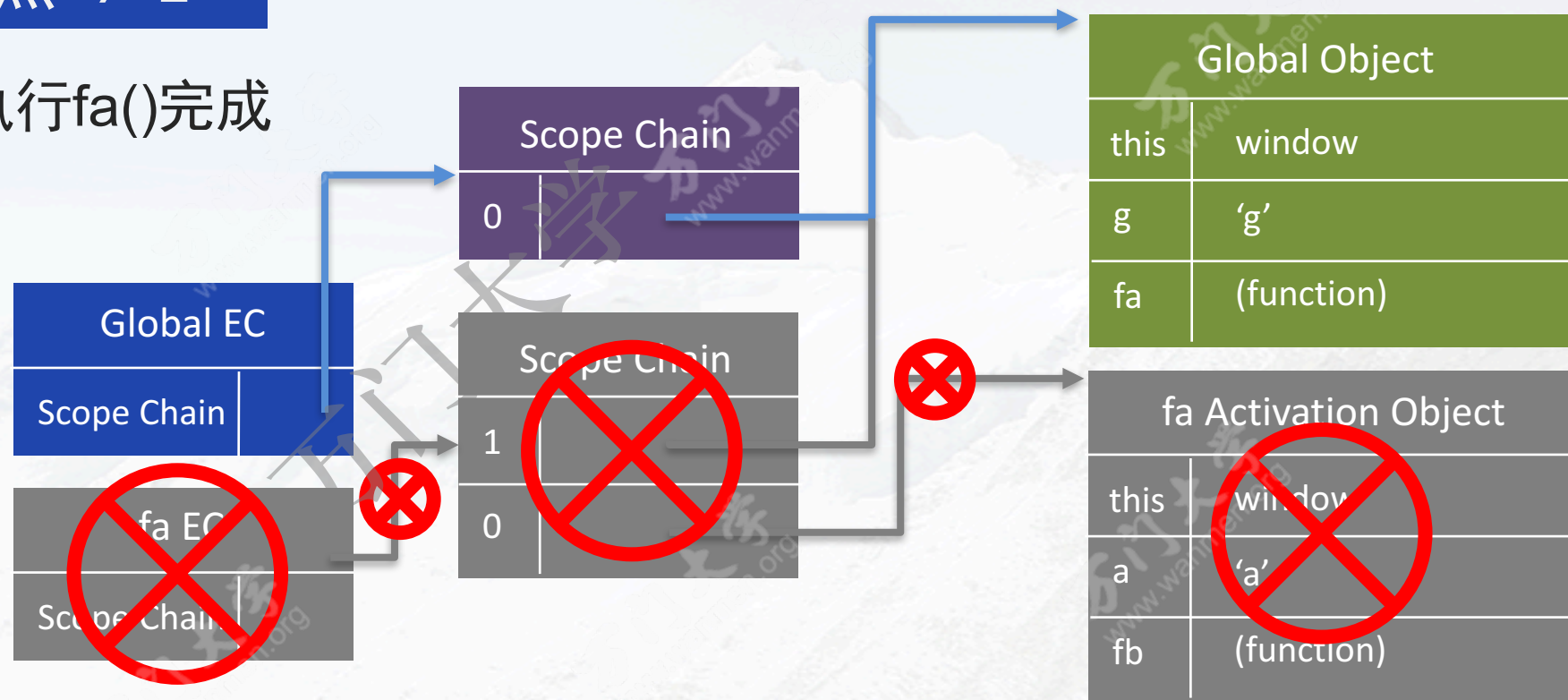


生成作用域链

知识点 9-2

✓ 5. fa-执行fa()完成

```
var g = 'g';
function fa(){
  var a = 'a';
  function fb(){
    var b = 'b';
  }
  fb();
}
fa();
```

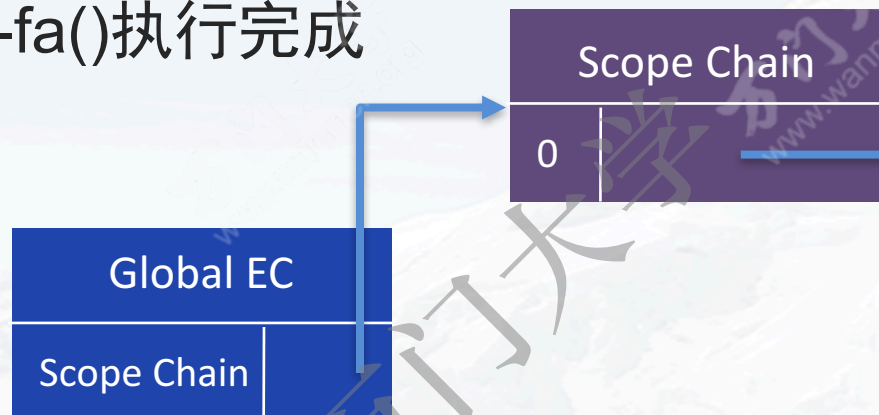


生成作用域链

知识点 9-2

✓ 6. 全局-fa()执行完成

```
var g = 'g';
function fa(){
  var a = 'a';
  function fb(){
    var b = 'b';
  }
  fb();
}
fa();
```



Global Object	
this	window
g	'g'
fa	(function)

生成作用域链

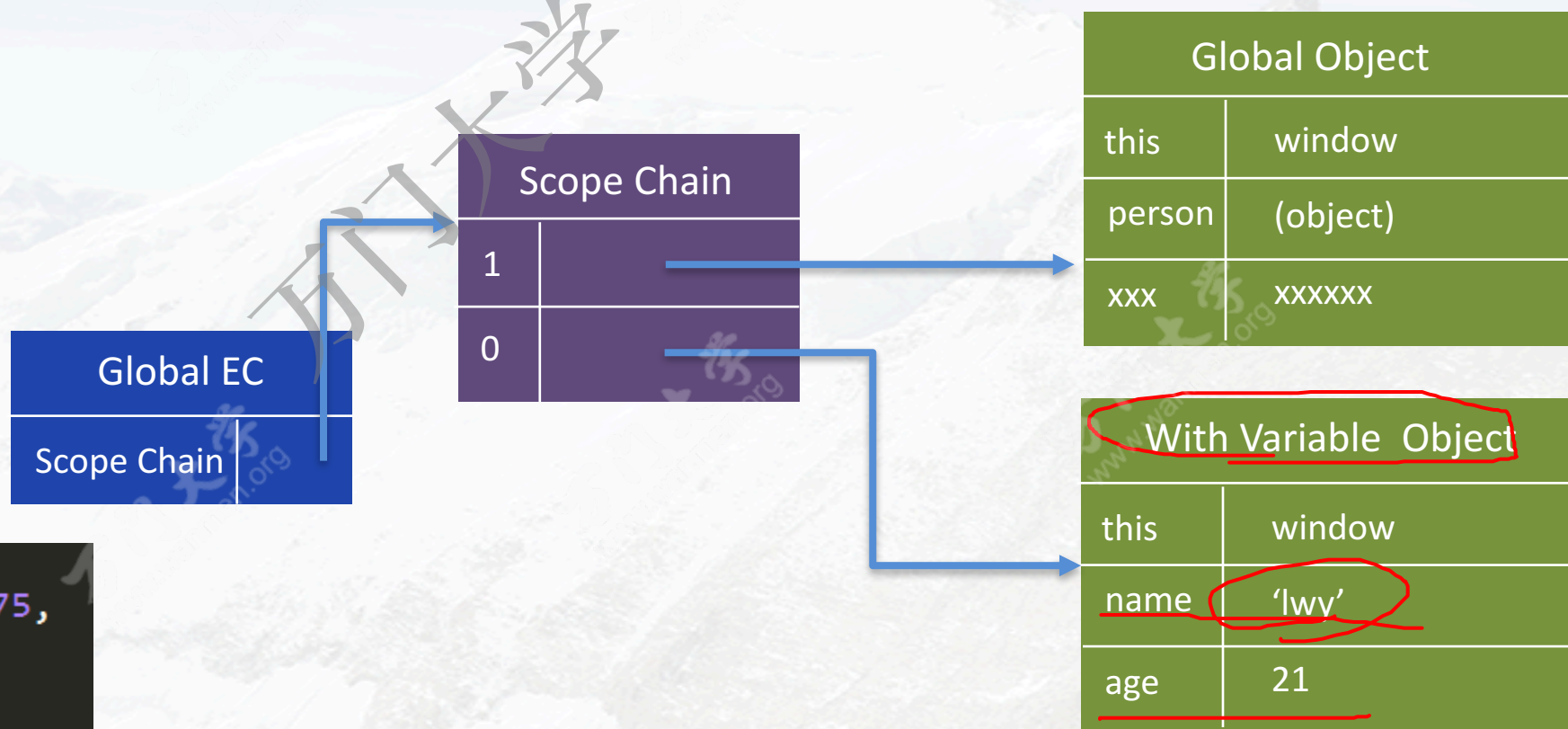
动手实验 9-2

- ✓ 函数多次调用时，是产生相同的AO还是不同的AO？
- ✓ 函数递归调用时，是产生相同的AO还是不同的AO？

生成作用域链

动手实验 9-2

- ✓ with, 生成新的with variable object, 放在作用域链表顶端。



```
person = {  
  name: "yhb", age: 22, height: 175,  
  wife: { name: "lwy", age: 21 }  
};  
with(person.wife){  
  console.log(name);  
}
```

生成作用域链

动手实验 9-2

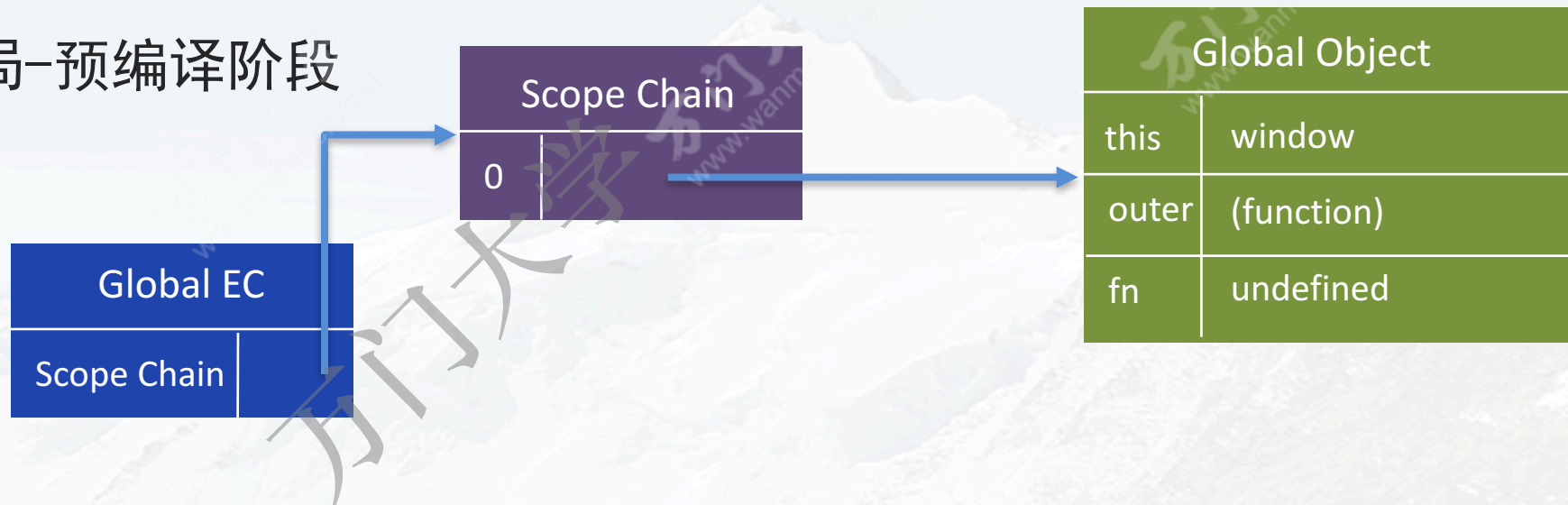
- ✓ 作用域链的应用：
 - ✓ 效率：
 - ✓ 尽量少使用靠近上层的变量，多使用自己的局部变量
 - ✓ 重名，容易出错：
 - ✓ 尽量减少不同层次函数使用相同的变量名
 - ✓ 避免函数名与变量名一样。
- ✓ 函数退出以后AO是否一定被释放？

闭包

动手实验 9-3

✓ 1. 全局-预编译阶段

```
function outer(){  
  var scope = 'outer';  
  function inner(){  
    return scope;  
  }  
  return inner;  
}  
var fn = outer();  
fn();
```

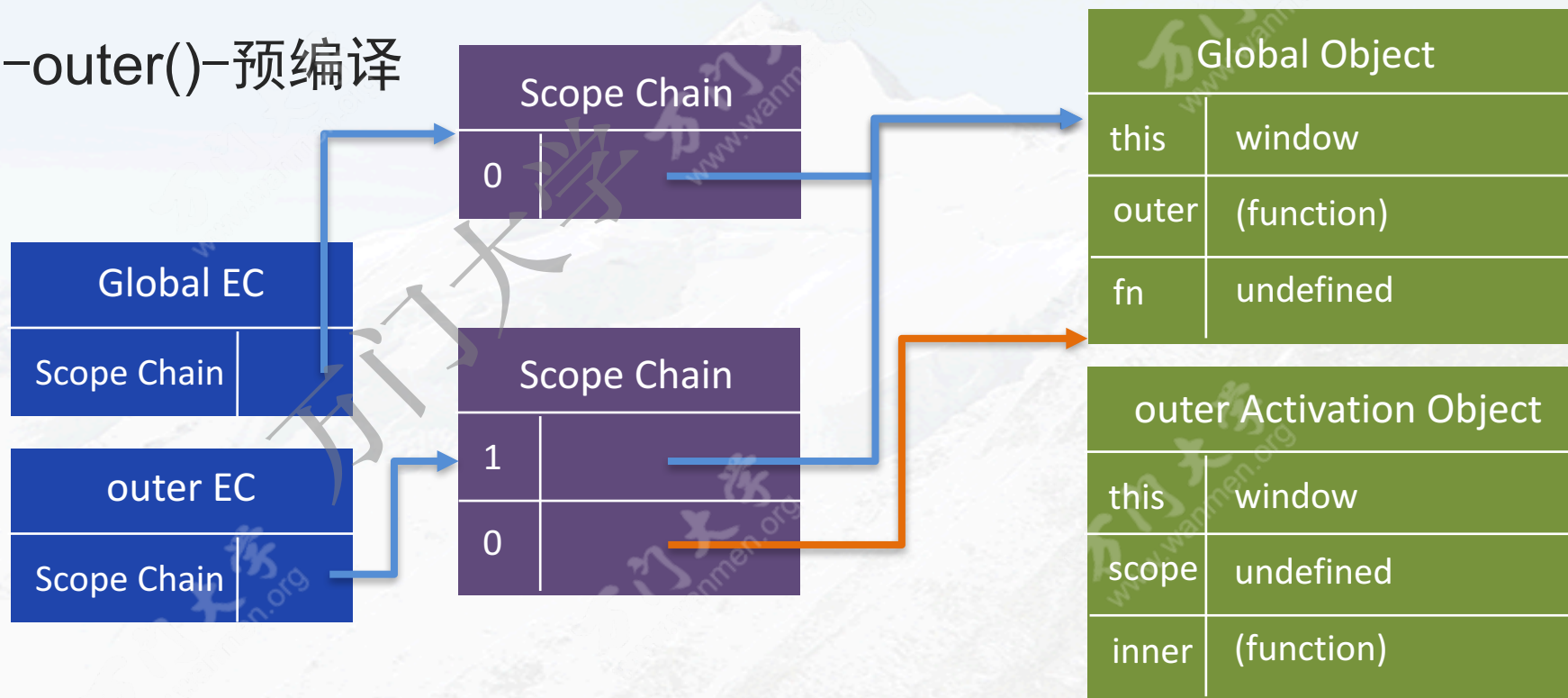


闭包

动手实验 9-3

✓ 2. 全局-outer()-预编译

```
function outer(){  
  var scope = 'outer';  
  function inner(){  
    return scope;  
  }  
  return inner;  
}  
var fn = outer();  
fn();
```



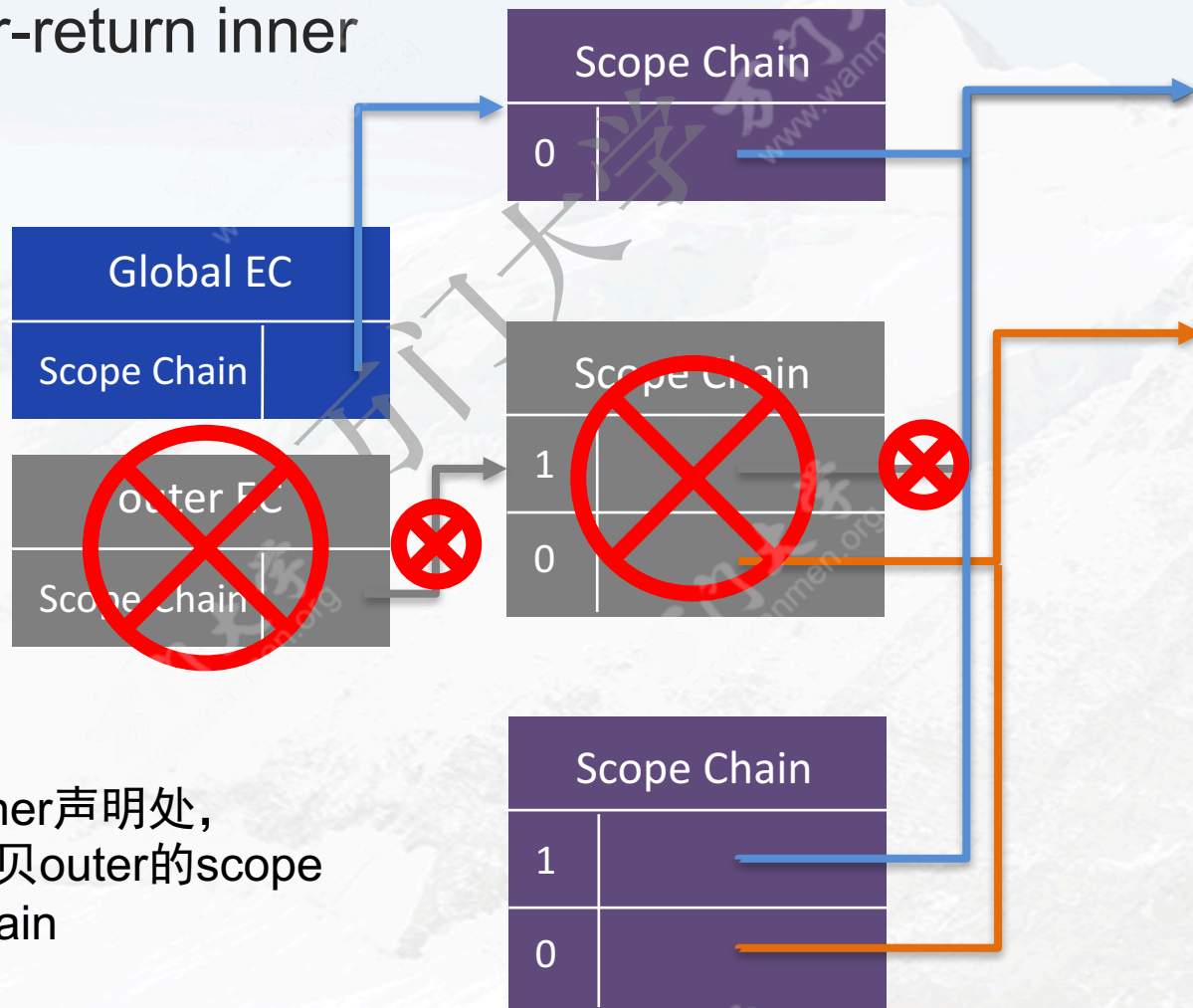
闭包

动手实验 9-3

✓ 3.outer-return inner

```
function outer(){  
  var scope = 'outer';  
  function inner(){  
    return scope;  
  }  
  return inner;  
}  
var fn = outer();  
fn();
```

Inner声明处，
拷贝outer的scope
chain



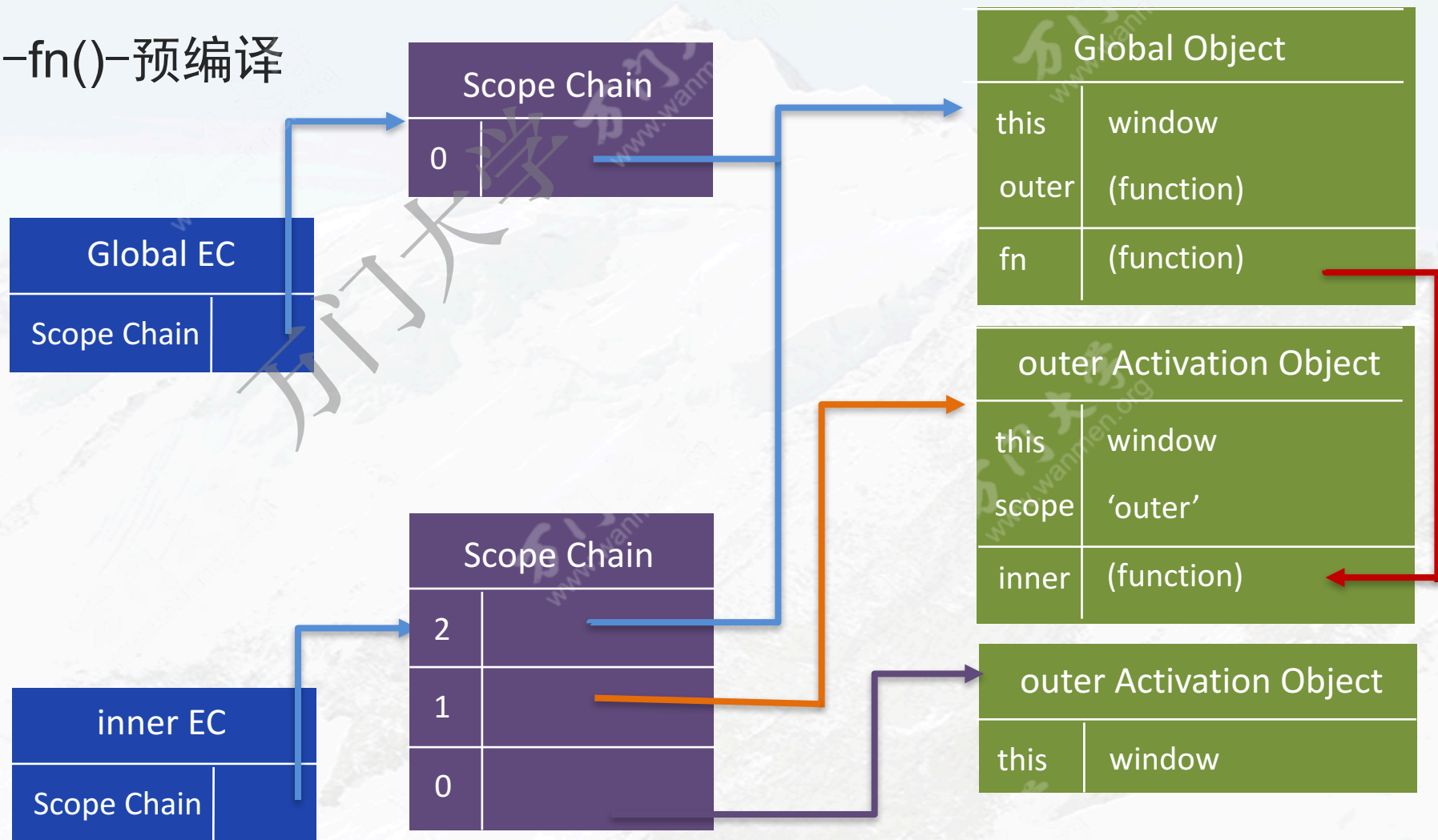
因为inner 拷贝的
scope chain继续指
向outer的AO，所以
outer的AO不释放

闭包

动手实验 9-3

✓ 4. 全局-fn()-预编译

```
function outer(){  
  var scope = 'outer';  
  function inner(){  
    return scope;  
  }  
  return inner;  
}  
var fn = outer();  
fn();
```



闭包

知识点 9-3

- ✓ 函数的AO通过scope chain相互连接起来，使得函数体内的变量都可以保存在函数的AO，这样的特性称为“闭包”。

动手实验 9-3

- ✓ 闭包的危险：
 - ✓ 闭包会造成原有AO不释放，产生内存泄漏
- ✓ 闭包的应用：
 - ✓ 实现公有变量
 - ✓ 缓存存储结构
 - ✓ 封装，实现属性私有化
 - ✓ 模块化开发，防止污染全局变量

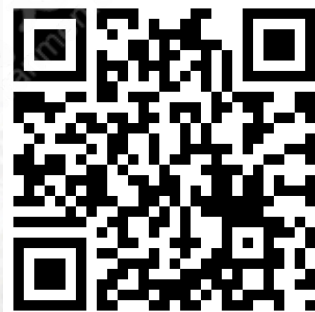
立即执行函数

动手实验 9-4

- ✓ 立即执行函数的优点：
 - ✓ 用完释放，减少内存压力

欢迎大家线下与我沟通

✓ 微信群



扫码此二维码进入分享群
可获得更多**书单**和**学习资料**