

您的潜力, 我们的动力

**Microsoft®**  
微软(中国)有限公司

# Microsoft AJAX Library中的 面向对象类型系统

赵劼

MSDN特邀讲师

[jeffz@live.com](mailto:jeffz@live.com)

# 本次课程内容包括

- 命名空间
- 类
- 接口
- 枚举
- 反射方法

# 收听本次课程需具备的条件

- 了解JavaScript

## Level 200

# 面向对象类型系统

- 合理使用面向对象编程
  - 提高复用性, 提高可维护性, etc.
  - 并非提高软件质量的保证
- 可以使用各种成熟的设计方式
  - 模式, etc
- 不拘泥于使用OO方式扩展对象
  - 结合JavaScript的动态特性
  - 下一次课程

# 命名空间

- 合理的组织大量的类型
  - 使开发人员能够方便地找到他们所需要的类型
  - 并非仅仅为了避免命名冲突
- 可重复注册
  - 每个独立的脚本模块前都要注册命名空间以保证命名空间存在

```
Type.registerNamespace("MyNamespace");
```



# 类

您的潜力, 我们的动力

**Microsoft**  
微软(中国)有限公司

- 定义步骤:
  - 定义构造函数
  - 定义成员（方法、属性、事件）
  - 注册类

# 类 – 构造函数

- 类的构造函数即为function定义
- 通常用于初始化域变量
- 私有成员使用下划线开头（无法真正封装）
  - this.\_myPrivateField;
  - this.\_myPrivateMethod();

# 类 – 构造函数

```
Type.registerNamespace("MyNamespace");
```

```
MyNamespace.MyClass = function(param1, ...) {  
    this._myField = null;  
    ...  
}
```



# 类 – 定义方法

- 基于prototype定义

```
MyNamespace.MyClass.prototype = {  
    myMethod1 : function() { ... },  
  
    myMethod2 : function() {  
        this.myMethod1(); // this保留字不可少  
    }  
}
```

# 类 – 定义属性

- Microsoft AJAX Library的面向对象类型系统将add\_和set\_开头的方法认作属性
- 避免定义只写属性, 使用某个方法替代

# 类 - 定义属性

```
MyNamespace.MyClass = function(param1, ...) {  
    this._myProperty = null;  
}  
MyNamespace.MyClass.prototype = {  
    get_myProperty : function() {  
        return this._myProperty;  
    },  
    set_myProperty : function(value) {  
        this._myProperty = value;  
    }  
}
```

# 类 – 注册类

```
MyNamespace.MyClass = function(param1, ...) {  
    // ...  
}  
MyNamespace.MyClass.prototype = {  
    // ...  
}  
MyNamespace.MyClass.registerClass(  
    'MyNamespace.MyClass');
```

您的潜力, 我们的动力

**Microsoft**<sup>®</sup>  
微软(中国)有限公司

# DEMO 1

## 定义及使用类



# 类 – 抽象类

- 包含抽象方法的类即为抽象类

```
MyNamespace.MyClass.prototype {  
    this._myMethod = function() {  
        throw Error.notImplemented();  
    }  
}
```

# 类 – 继承

- 调用父类的构造函数
  - 没有父类的类可以省略这步
  - 有父类的类必须调用父类的构造函数, 否则会丢失继承效果

```
MyNamespace.MyClass = function(){  
    MyNamespace.MyClass.initializeBase(this, [param1, ...]);  
    // ...  
}
```

# 类 – 继承

- 注册类时提供父类
- 可以直接以普通方式实现父类的抽象成员
- `toString`方法无法被继承

```
MyNamespace.MyClass.registerClass(  
    'MyNamespace.MyClass'  
    MyNamespace.MyParentClass);
```

# 类 – 调用父类方法

```
MyNamespace.MyClass.prototype = {  
  myMethod : function(param1, ...) {  
    MyNamespace.MyClass.callBaseMethod(  
      this, "myMethod", [param1, ...]);  
  },  
  anotherMethod : function() {  
    MyNamespace.MyClass.callBaseMethod(  
      this, "myMethod", [...]);  
    this.myMethod(...);  
  }  
}
```

您的潜力, 我们的动力

**Microsoft**<sup>®</sup>  
微软(中国)有限公司

# DEMO 2

## 抽象类与继承



# 接口

- 与类的定义方法大致相同
- 构造函数抛出异常
- 所有的方法抛出异常
- 注册接口时使用registerInterface方法
- 接口无法继承其他接口
- 由于JavaScript为弱类型语言, 因此Microsoft AJAX Library中接口的作用只是标记, 可以从某些反射方法中反映出效果

# 接口

您的潜力, 我们的动力

**Microsoft**  
微软(中国)有限公司

```
MyNamespace.IMyInterface = function() {  
    throw Error.notImplemented();  
}  
  
MyNamespace.IMyInterface.prototype = {  
    myMethod : function() {  
        throw Error.notImplemented();  
    }  
}  
  
MyNamespace.IMyInterface.registerInterface(  
    "MyNamespace.IMyInterface");
```

# 接口 – 实现接口

- 在使用**registerClass**方法注册类时可以传入额外的参数来实现接口

```
MyNamespace.MyClass.registerClass(  
    "MyNamespace.MyClass",  
    null, // 表示没有父类  
    MyNamespace.IMyInterface1,  
    MyNamespace.IMyInterface2,  
    ...  
);
```

您的潜力, 我们的动力

**Microsoft**<sup>®</sup>

微软(中国)有限公司

# DEMO 3

## 定义及使用接口

# 枚举

您的潜力, 我们的动力

**Microsoft**  
微软(中国)有限公司

- 枚举即为Number
- 增加可读性
- 可以定义为标记 (Flag)
- 每个枚举类型均有toString和parse方法



# 枚举

您的潜力, 我们的动力

**Microsoft**  
微软(中国)有限公司

```
MyNamespace.MyEnum = function() {  
    throw Error.notImplemented();  
}  
MyNamespace.MyEnum.prototype = {  
    Item1 : 1,  
    Item2 : 2,  
    Item3 : 4  
}  
MyNamespace.MyEnum.registerEnum(  
    "MyNamespace.MyEnum" [, true]);
```

# DEMO 4

## 定义及使用枚举

# 反射方法

- `Type.prototype.getBaseType`
- `Type.prototype.getInterfaces`
- `Type.prototype.getName`
- `Type.prototype.implementsInterface`
- `Type.prototype.inheritsFrom`
- `Type.prototype.isImplementedBy`
- `Type.prototype.isInstanceOfType`、
- `Type.getRootNamespaces`

# 反射方法

- `Type.isClass`
- `Type.isInterface`
- `Type.isNamespace`
- `Type.isEnum`
- `Type.isFlags`
- `Type.parse`
- `Object.getType`
- `Object.getTypeName`


# 获取更多MSDN资源

- **MSDN中文网站**  
<http://www.microsoft.com/china/msdn>
- **MSDN中文网络广播**  
<http://www.msdnwebcast.com.cn>
- **MSDN Flash**  
<http://www.microsoft.com/china/newsletter/case/msdn.aspx>
- **MSDN开发中心**  
<http://www.microsoft.com/china/msdn/DeveloperCenter/default.mspix>




# Question & Answer

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。

 **问题和解答 (无问题)** ▲ ×

在此会议中尚未解答任何问题。

要向演示者提问，请在此处键入问 

提问(A)

删除(D)

问题管理器(Q)

您的潜力, 我们的动力

**Microsoft®**  
微软(中国)有限公司

**Microsoft®**