您的潜力，我们的动力

Microsoft
微软(中国)有限公司

JavaScript的原生类型以及Microsoft AJAX Library的相关扩展（上）

赵劼

MSDN特邀讲师

jeffz@live.com

本次课程内容包括

- Microsoft AJAX Library定义
- JavaScript中以下原生类型的使用及扩展
  - Object
  - Array
  - Error
  - Function
- 讲解原生类型最常使用的功能，并非完全分析。

下次课程内容包括

- JavaScript中以下原生类型的使用及扩展
  - Boolean
  - Number
  - Date
  - String

课程中不会涉及的原生类型

- JavaScript中以下原生类型的使用及扩展
  - RegExp
  - Undefined
  - Null
  - EvalError
  - etc.

收听本次课程需具备的条件

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

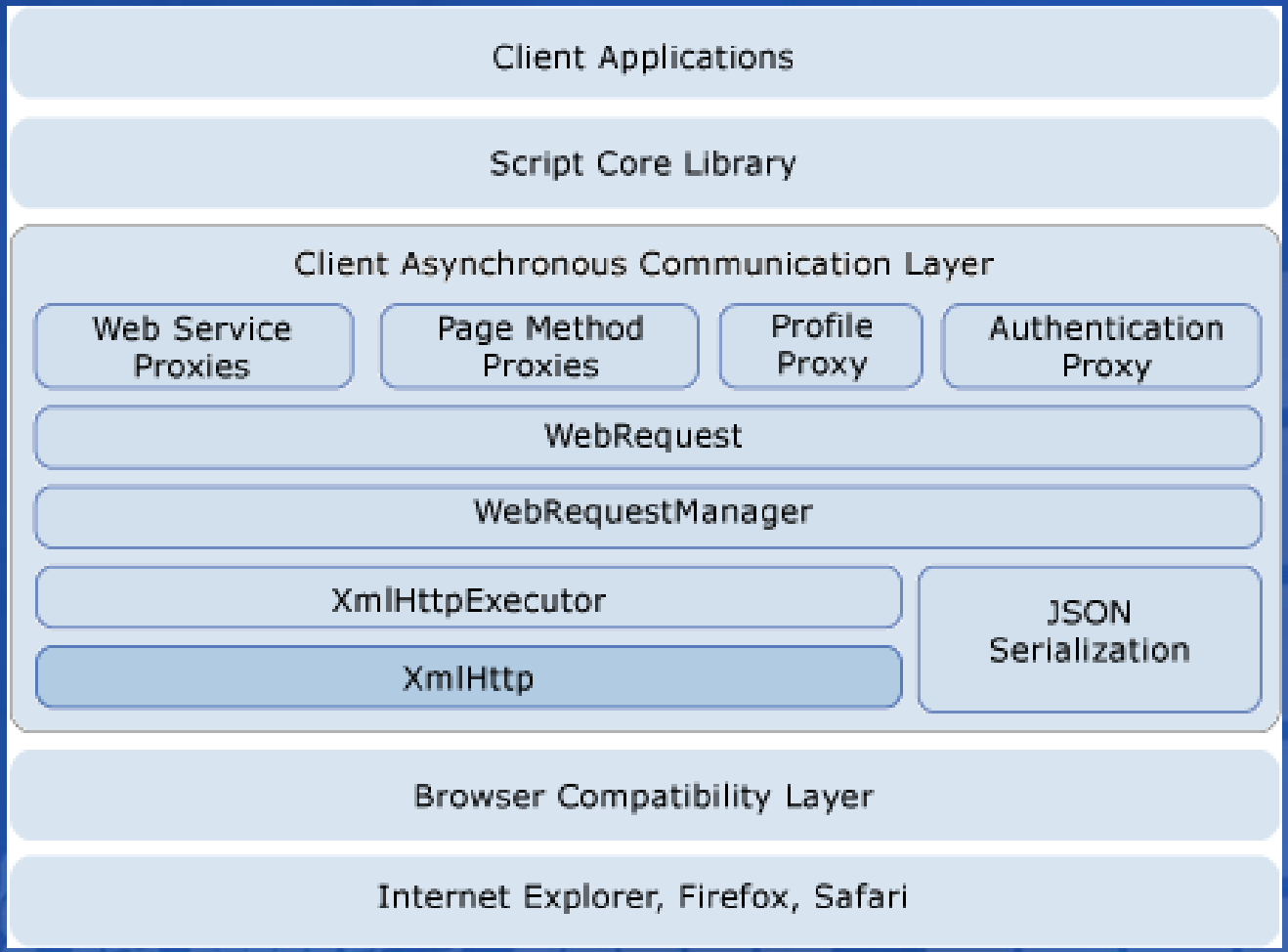
- 了解JavaScript基本开发

Level 200

什么是Microsoft AJAX Library？

- ASP.NET AJAX的客户端部分
- 纯客户端框架
  - ASP.NET AJAX能够独立于各服务器技术的官方理由
- 提供了JavaScript扩展和基础类库

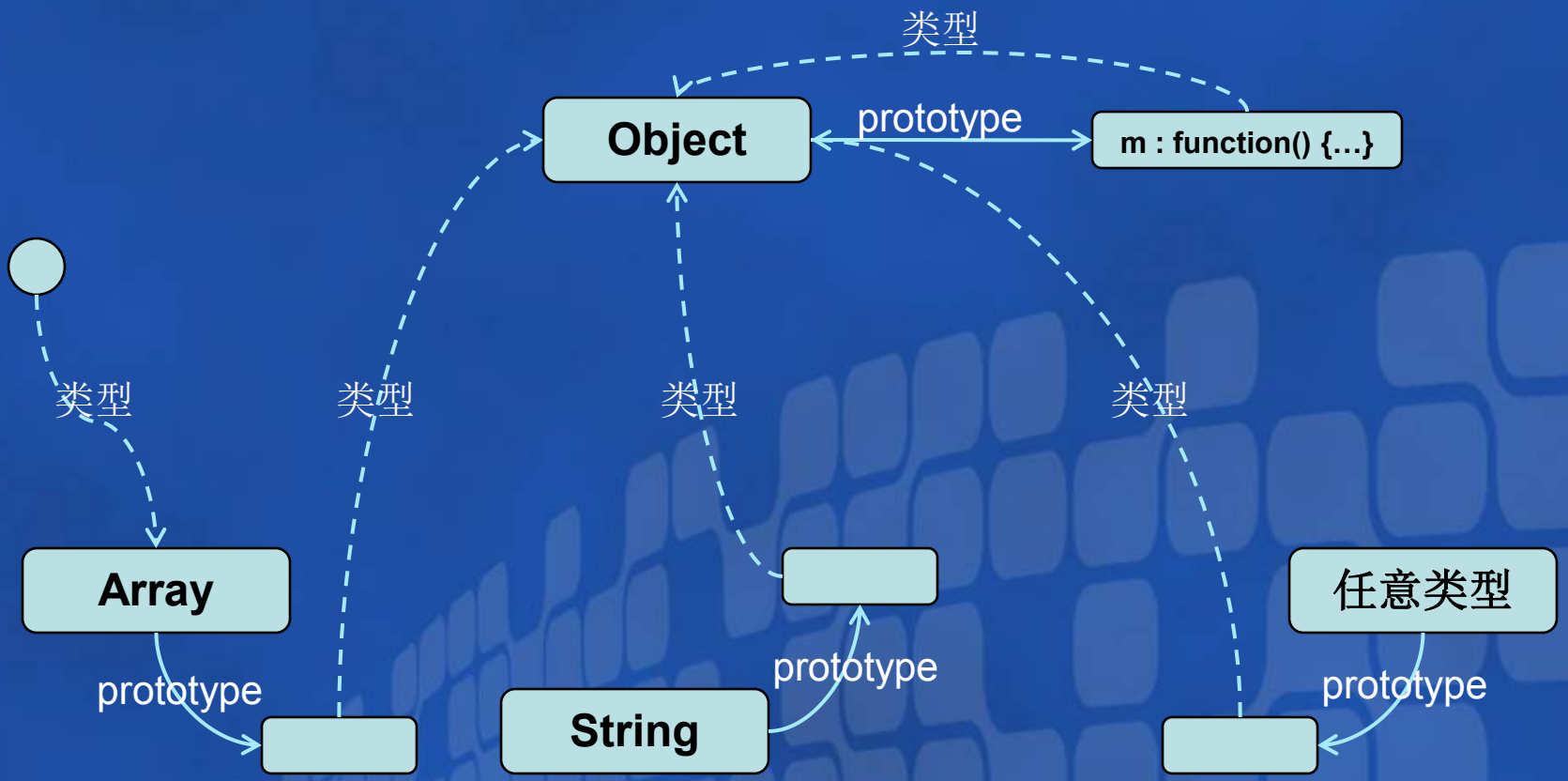
Microsoft AJAX Library组成



Object原生类型

- ECMAScript Spec中定义（ecma-262）
  - 一个无序的集合，可以存放任意类型对象
- 常作为字典使用
  - o.ItemName
  - o[“ItemName”]
- 可以使用for - in遍历字典中每一项
- 禁止扩展其prototype对象
  - 其扩展将会出现所有的对象中
  - 影响for - in操作的结果
- Microsoft AJAX Library并没有扩展Object类型

Prototype链模型



Object.prototype. …

- 出现在所有对象中（原因见上图）

- toString() / toLocaleString()
  - 得到表示当前对象与环境无/有关的字符串
- valueOf()
  - 返回表示该对象的value（大部分类型会覆盖这个方法）
- hasOwnProperty(propertyName)
  - 对象上是否直接定义了某个属性
  - 不考虑prototype链
- isPrototypeOf(obj)
  - obj是不是当前对象的prototype对象
  - 顺着prototype链查找
- propertyIsEnumerable(propertyName);
  - 某属性是否可遍历
  - 不考虑prototype链

DEMO 1

使用Object原生类型

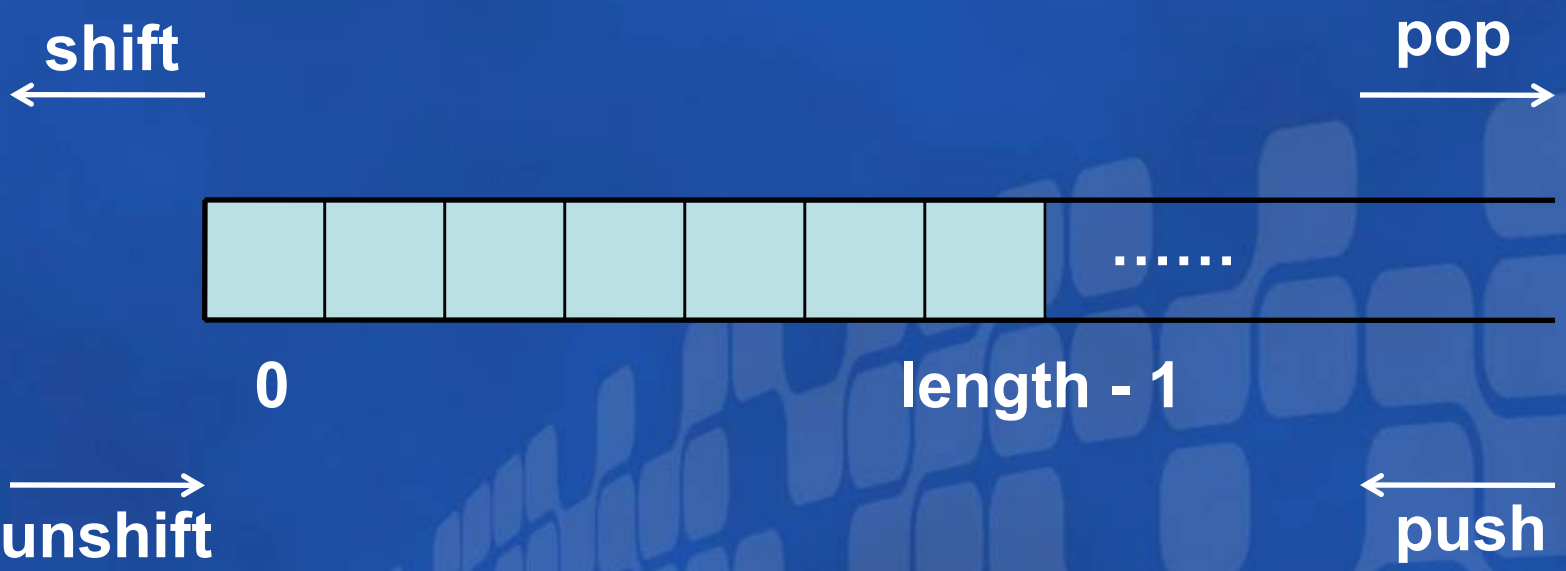
Array原生类型

- 长度可变数组对象
  - 0-based
  - 最大长度2^32−1 = 4294967295
- 构造Array对象
  - new Array()：空数组
  - new Array(3)：长度为3的数组
  - new Array(1, "Hello")：构造有两个元素的数组

Array.prototype. …

- length属性：数组长度，可读写
- toString() / toLocaleString()方法：
  - 返回逗号分割的字符串
  - 两者区别是得到每个对象字符串的方式
- concat([item1 [, item2 [, …]]])方法：
  - 返回一个新数组，保存了原数组所有元素和所有的参数
- push([item1 [, item2 [, …]]])方法：
  - 在数组尾添加一个或多个元素
- pop方法：
  - 从数组尾去除并返回元素
- shift方法：
  - 从数组头去除并返回元素
- unshift([item1 [, item2 [, …]]])方法：
  - 在数组头添加一个或多个元素

Array.prototype. …(cont.)



Array.prototype. …(cont.)

- join(separator)方法：
  - 返回以separator作为分割符得到一个连接所有元素的的字符串
  - StringBuilder的基础，可以为IE加快字符串拼接速度
- reverse()方法：将数组内所有元素逆转
- sort(compareFunction)方法：
  - 参数为一个方法，用于比较两个元素
  - 省略了参数则直接使用<, ==, >比较两个元素

Array.prototype. …(cont.)

- slice(start, end)方法：
  - 返回新数组，不影响旧数组
  - 包含从下标start开始到下标end-1的元素
  - 如果省略end则包含从下标start开始至末尾的所有元素
  - 如果参数为负数，则表示数组的“倒数”第几个下标（即下标为n + array.length）
  - 如果start元素在end元素之后或相同，则返回空数组

Array.prototype. …(cont.)

- splice (start, deleteCount [, item1 [, item2 [, …]]]) 方法：
  - 最灵活的方法，影响当前数组
  - 从下标start的元素开始，删除deleteCount个元素，并在当前start位置开始插入剩余元素
  - 删除元素：splice(2, 1)
  - 插入元素：splice(2, 0, "Hello", "World")
  - 替换元素：splice(2, 1, “Hello”, “World”)

DEMO 2

使用Array原生类型

Array原生类型的扩展

- 全都是静态方法
  - 为了和其他类库兼容（Prototype）
- 提供了一些常用的方法
- 提供了语义良好的方法名
- 大多数方法为简单封装

Array原生类型的扩展（cont.）

- Array.enqueue(array, item)：
  - “入队列”操作，将item添加至array末尾。
- Array.dequeue(array)：
  - “出队列”操作，返回并删除array的第一个元素。
- Array.addRange(array, items)：
  - 将items数组中所有元素添加至array末尾。
- Array.contains(array, item)：
  - 如果array中包含item元素，则返回true，否则返回false。
- Array.clear(array)：
  - 清除array中的所有元素。

Array原生类型的扩展（cont.）

- Array.insert(array, index, item)：
  - 将item插入至array中下标为index的位置。
- Array.remove(array, item)：
  - 从array中移除item元素。
- Array.removeAt(array, index)：
  - 从array中移除下标为index的元素。
- Array.clone(array)：
  - 返回一个与array相同的新数组。
- Array.parse(value)：
  - 将表示数组的JSON字符串变为一个数组对象。

Array原生类型的扩展（cont.）

- Array.indexOf(array, item, start)：
  - 获得item在array中的下标，从下标为start开始查找。如果array中没有item元素，那么返回-1。
- Array.add(array, item)：
  - 将item添加至array末尾，它和Array.enqueue其实是同一个函数。
- Array.forEach(array, method, instance)：
  - 以instance为上下文this引用，将array中的每个元素依次作为参数，循环调用method方法。

DEMO 3

使用Array.forEach方法

Error原生类型

- 表示错误对象
  - EvalError, URIError, RangeError, etc.
- 捕获方式：
  - try { … throw new Error(…) } catch(e) { … }
  - 理论上可以throw出任意对象
- Error对象IE和FireFox公有属性
  - message：错误信息

Error浏览器特定属性

- IE：
  - description：同message属性
  - number：错误编号，只有脚本引擎抛出的错误才有该属性
- FireFox：
  - fileName：创建错误的文件
  - lineNumber：创建错误对象的行号
  - stack：创建错误时的堆栈信息

DEMO 4

抛出、捕获、并显示错误

Error原生类型的扩展

- Error.create(message, errorInfo)方法：
  - 创建新的Error对象
  - 将Error对象的错误信息属性设为message
  - 将errorInfo上的信息附加到Error对象
- Error.prototype.popStackFrame()方法：
  - 为Error对象整理出更优雅直观的信息（lineNumber, stack）
  - 对于IE无效
  - 如果一个方法仅仅是返回Error对象而不是抛出对象，则在返回前应该调用该方法

DEMO 5

使用Error原生类型的扩展

Function原生类型

- 与Array，String等类型处于同等地位
- 每个方法均为Function类型的实例
  - typeof(Array) == typeof(Function) == “function”
- 方法调用时根据发起的对象来确定this上下文引用
- Function.prototype.apply(instance, args)
- Function.prototype.call(instance, [arg1 [, arg2 [, …]]])

DEMO 6

使用Function原生类型

Function原生类型扩展

- Function.createDelegate(instance, method)方法：
  - 得到一个方法引用，执行它时则会调用method方法，并且保证method方法的上下文this引用为instance
- Function.createCallback(method, context)方法：
  - 得到一个方法引用，执行它时则会调用method方法，并将context作为额外的参数传入

DEMO 7

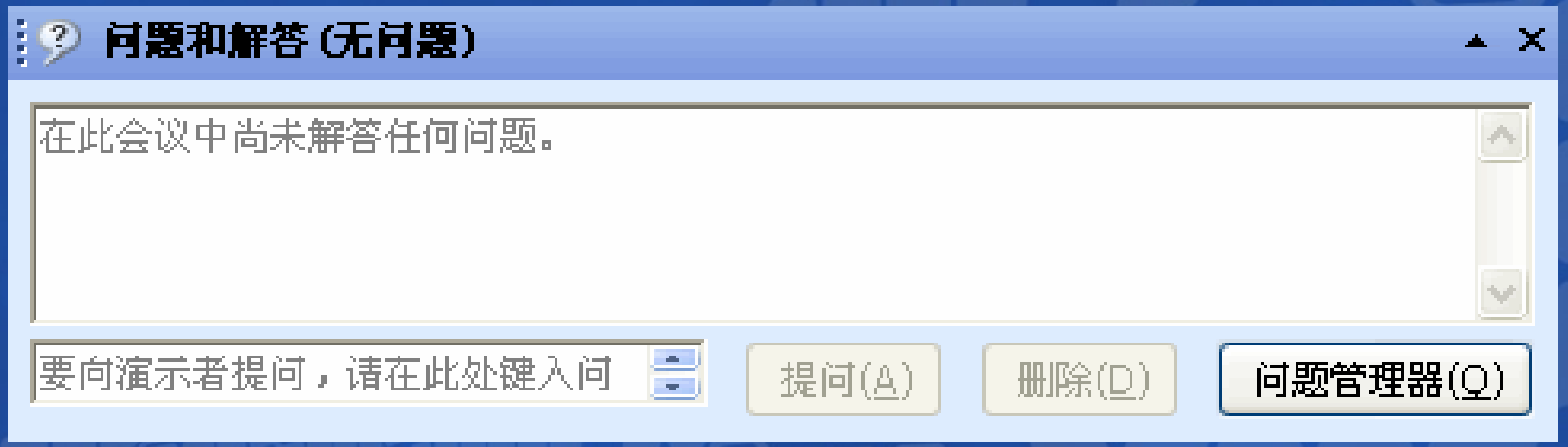
使用Function原生类型及扩展

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.mspx

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。



您的潜力，我们的动力

Microsoft®
微软(中国)有限公司

msdn

MSDN Webcasts