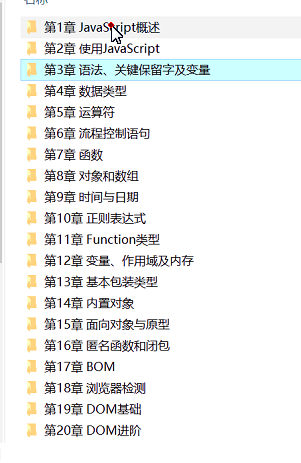
//盖博 麦秀 公主 米林 .... 李珍 ... 玫瑰 ... ... 美丽

# JS：单线程语言



## 交互、表单验证、特效

函数对象：

Function box(){}

Var box = funxtion(){}

Var a = new Function(a,b){‘return a+b’}

普通对象

New object

## js核心

* 1. 核心：ECMAScript
  2. 文档对象模型：DOM
  3. 浏览器对象模型：BOM

## 语法构成

* 1. 区分大小写
  2. 标识符
  3. 直接量
  4. Typerof：判断数据类型

## 变量

## 数据类型

String、number、null、object、boolean、undefined

typeof：能检测出是什么类型

* 1. String：字符串
  2. Boolean：布尔值
  3. Number：数值
  4. Undefined：未定义
  5. Null：
  6. Object：对象或null
  7. Function：函数
  8. **空字符串，null，undefined，0，，转换成boolean是false**
  9. **空数组‘[ ]’，数组{ } 可以转换成boolean的true**

## 运算符-表达式

* 1. 对象会自动调用tostring()方法和valueOf()
  2. 只要有NaN都是NaN
  3. 100-null=0(null转换成0)
  4. 两个 ‘！！’转化之后表示boolean的true

## 流程控制语句

* 1. If()else{}
  2. For(){}
  3. While(){}

## 函数

## 对象和数组

* 1. 创建object
     1. 使用字面量
     2. New(可省略)
  2. 属性输出方式
     1. alert(box.age);
     2. alert(box['age']);
  3. 创建数组两种方式
     1. New运算符
     2. 字面量
  4. 栈方法：先进后出
     1. Pop()方法用于删除并返回数组的最后一个元素(把数组长度减1，并返回所删除元素的值，若数组为空，则不改变数组，并返回undefined值)
     2. Push：逐个添加到数组的末尾，
        1. alert(push(a))--打印并返回修改后数组的长度
  5. 堆方法：先进先出
     1. **Shift()**：移出数组开头元素，
        1. alert(shift(a))--打印并返回移出的元素
     2. **Unshift()**和push相反，为数组的前端添加一个元素
     3. **Splice(a,b)：删除元素，**a:删除开始位置 b:删除数量
     4. **Slice()：**从已有的数组中返回选定的元素(可以使用负值从数组的尾部选取元素)
     5. **Concat：**合并两个字符串
  6. Alert(box.push(‘郑州’))---数组末尾添加一个元素，并且返回长度
  7. 数组排序：reverse()---逆向排序；和sort()---从小到大排序
     1. Sort：要是数字，要先设置一个函数，要不然会有一些问题，
     2. **function f(a,b){return (a-b)}**
  8. 方法是有返回值的，直接写在里面是方法
  9. Arguments是动态参数，有参无参是固定

## 时间与日期：date

* 1. date.parse()==getTime---转换成时间戳
  2. getTime得到13位时间戳

## 正则表达式

* 1. 两种创建方式：字面量，new
  2. 参数：
     1. I:忽略大写
     2. M：多行匹配
     3. G：全局匹配
  3. text:
     1. 匹配返回true，反之返回false
  4. exec：
     1. 匹配返回匹配值，反之为null
  5. String提供方法
     1. match：返回pattern中的子串或null
     2. search：查到第一个出现的位置并返回
     3. replace(a,b)：查找之后，并把值替换(b-->a)
     4. split：把字符串变成变成数组
        1. str.split(pattern)：将空格拆开分成数组
  6. 获取控制
     1. 。。
     2. 贪婪模式：尽可能多的匹配（加 ？ 关闭贪婪匹配）

## Function类型

* 1. 两个特殊对象属性
     1. arguments：类数组对象
        1. arguments.callee：代替本身
     2. This
  2. 函数属性和方法
     1. length：表示函数接收的参数的个数
     2. Prototype
        1. 实际上就是对象冒充
        2. **apply()**：调用一个对象的方法，用另一个对象替换当前对象

eg：**B.apply(A，arguments)**-->即A对象应用B对象的方法

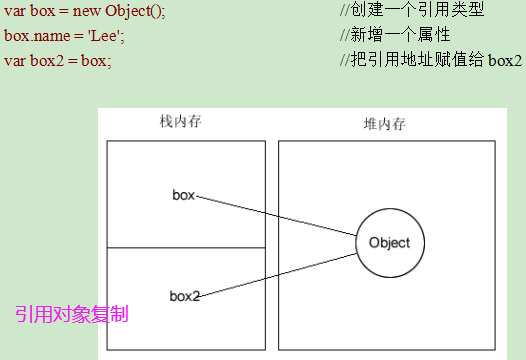
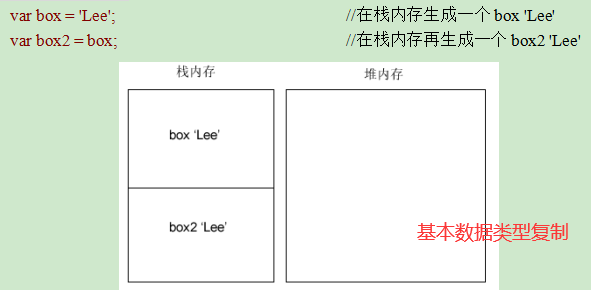
* + - 1. **call()**：调用一个对象的一个方法，用另一个对象替换当前对象

eg：**B.call(A，args1，args2)**-->即A对象调用B对象的方法

* + 1. apply和call区别
       1. 都是调用一个对象的一个方法，用另一个对象替换当前对象
       2. **不同之处 ：**传递的参数：
          1. apply最多只能有两个参数--新this对象和一个数组，若不是数组会报错
          2. Call可以传递多个参数，第一个参数和apply一样，是用来替换的对象，后边是参数列表
       3. **要是知道明确的参数使用call，反之 用apply**

## 变量、作用域、内存

* 1. var 变量提顶
  2. 闭包就是将局部变量存在内存中，能连续调用
  3. 动态属性：引用对象能随意添加属性，基本数据类型不能
  4. 复制变量值
     1. 基本数据类型复制的是值本身
     2. 引用类型复制的是地址

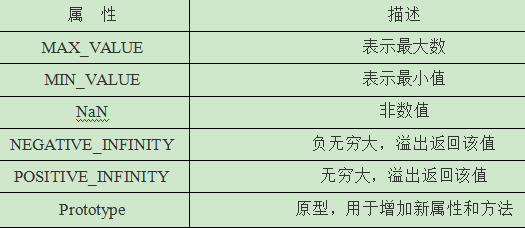


* 1. 传递参数
     1. 所有函数都是按值传递的
     2. 参数不会按引用传递
     3. 函数的参数都是局部变量
  2. 检测类型：typeOf
     1. 通过typeOf运算符类判别
        1. 只能判断出object和function，判断不出精确数据类型
     2. 可使用**instanceof**
  3. 执行环境及作用域
     1. 全局执行环境被认为是window对象
     2. 全局的变量和函数都是window对象的属性和方法
  4. 块级作用域
     1. 块级作用域：表示诸如if语句等有花括号封闭的代码块
     2. **没有作用域：if、for、**函数中的var关键字
  5. 内存问题

## 基本包装类型

**三个特殊测引用类型：boolean、number、string**

* 1. Substring：截取指定位置的字符串(不包含这个数字)
  2. Number：有一些静态属性（直接调用属性，不用new运算符）和方法
     1. 静态属性：



* + 1. Number对象的方法
       1. **tostring()**：将数值转化为字符串，并且可以转换进制
       2. **toFiexd()**：将数字保留小数点后指定位数并转化为字符串
          1. 还有四舍五入功能（var a = 100000.789;）
       3. toLocaleString()：根据本地数字格式转换为字符串
       4. toExponential()：将数字以指数形式表示，保留小数点后制定位数并转化为字符串
       5. toPrecision()：指数形式或点形式表述数，保留小数点后面指定位数并转化为字符串
    2. String类型
       1. 属性
          1. length：返回字符串的字符长度
          2. constructor：返回创建string对象的函数
          3. prototype：通过添加属性和方法扩展字符串定义
       2. 方法
          1. **charAt(n)**：返回指定索引位置的字符
          2. **charCodeAt(n)**：以unicode编码形式返回指定索引位置字符
          3. valueOf()、toLocaleString()、toString():返回串的基本值

## 内置对象

* 1. Global对象:包含一些**属性**：undefined、nan、object、array、function
     1. **并不存在全局变量和全局函数；所有在全局作用域定义的变量和函数，都是Global对象的属性和方法**

**任何不属于某个对象的属性和方法都是全局的**

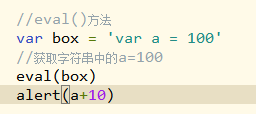
* + 1. **方法**
       1. **URI编码方法**
          1. encodeURI()：编码

encodeURIComponent():彻底编码

* + - * 1. decodeURI():解码

decodeURIComponent():彻底解码

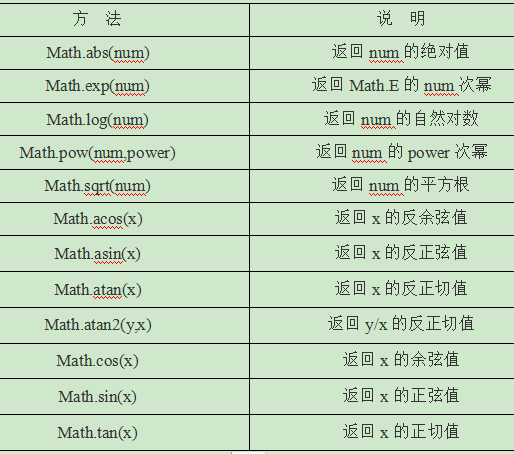
* + - 1. **eval()**
      2. 相当于一个字符串解析器
      3. 获取字符中的表达式



* 1. Math对象
     1. min()和max()
        1. Math.min():用于确定一组数值中最小的值
        2. Math.max()用于确定一组数值中最大的值
     2. 舍入方法
        1. ceil():向上取整
        2. floor()：向下取整
        3. round()：四舍五入
     3. random()
     4. 属性：静态属性、属性
     5. Math对象的属性



* + 1. 其他

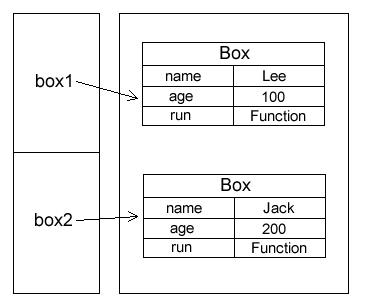


## 面向对象与原型

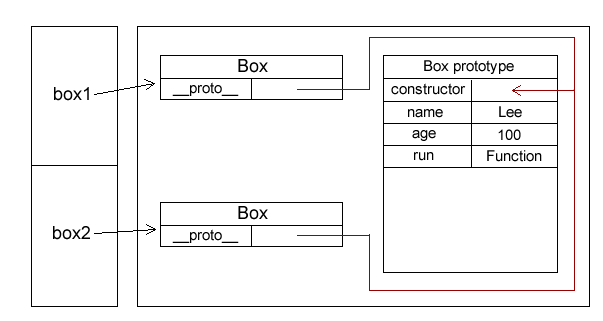
* 1. 工厂模式
     1. 缺点：
        1. 代码冗余
        2. 不明确指定对象
  2. 构造函数

解决对象识别问题

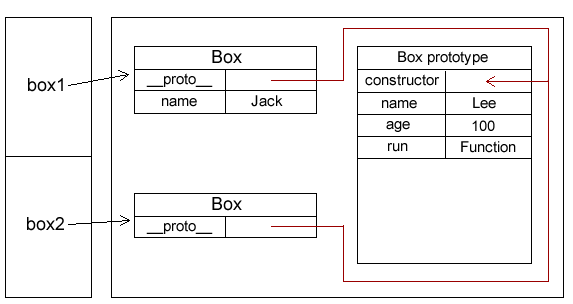
* + 1. 规范
       1. 函数名和实例化构造名相同相同且大写
       2. 通过构造函数创建对象，必须使用new运算符
  1. 原型：共享属性和方法
     1. 原型模式的**执行流程**
        1. 先查找构造函数实例里的属性或方法，若有，立即返回
        2. 若构造函数实例里无，则去他的原型对象里找，若有，返回
     2. 共享属性和方法
     3. 构造函数



* + 1. 原型模式方式



* + 1. **构造函数实例属性和原型属性示意图**



* + 1. **\_\_proto\_\_:**实例指向原型对象的一个指针
       1. 作用：指向构造函数的原型属性**constructor**
    2. **判断一个对象是否指向该构造函数的原型对象，使用isPrototypeOf()方法来测试**
       1. 下面这句**构造函数的原型(Box)是不是实例(box)的原型**

**alert(Box.prototype.isPrototypeOf(box))**

* + 1. **使用hasOwnProperty()函数判断属性是在构造函数里还是在实例里**

**console.log(box.hasOwnProperty(‘name’)):**

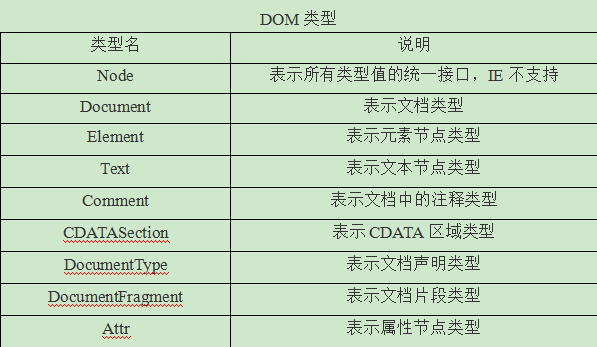
**实例中有返回true，反之返回false**

* + 1. **in操作符：**可以判断实例或原型中是否存在属性

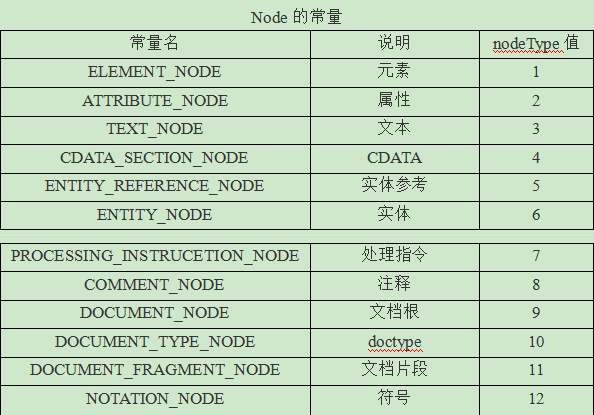
**Console.log(‘name’in box3)**

* + 1. **构造函数和字面量创建**基本相同
       1. **字面量创建的方式使用constructor属性不会指向实例，而会指向object**
       2. **构造函数创建的方式则相反**
  1. **继承**
     1. **原型继承**
     2. **对象冒充继承**
     3. **构造函数继承：**（借用构造函数+原型链）
     4. **原型式继承**

1. **没写**
2. **没写**
3. **没有**
4. **DOM基础**
5. **DOM进阶**
   1. **DOM类型**



* 1. **node的常量**



* 1. **Children**
  2. **关系掩码表**

1. **没**
2. **DOM元素尺寸和位置**
   1. **获取元素CSS大小**
      1. 通过style内联获取元素大小
      2. 通过计算获取元素大小
      3. 通过CSSStyleSheet对象中的cssRules(或rules)属性获取元素大小(cssRules只能获取到内联和链接样式的宽和高，不能获取到行内和计算后的样式)
   2. **获取元素实际大小**
      1. **clientWidth和clientHeight**

获取元素可视区的大小，得到元素内容及内边距锁占据的空间大小

* + 1. **scrollWidth和scrollHeight**

可以获取滚动内容的元素大小

* + - 1. 增加内边距，最终值会等于原本大小加上内边距大小
      2. 增加滚动条，最终值会等于原来大小减去滚动条大小
      3. 增加外边距，无变化
    1. **offsetWidth和offsetHeight**

可以返回元素实际大小，包含边框、内边距和滚动条

* + - 1. 增加边框，最终值等于原本大小加上边框大小
      2. 增加内边距，最终值会等于原本大小加上内边距大小
      3. 增加外边距，无变化
      4. 增加滚动条，无变化
  1. **获取元素周边大小**
     1. **clientLeft和clientTop**

可以获取元素设置了左边框和上边框的大小

* + 1. **offsetLeft和offsetTop**

可以获取当前元素相对于父元素的位置

* + 1. **scrollTop和scrollLeft**

可以获取滚动条被隐藏的区域大小，也可以设置定位到该区域

1. 没
2. **事件入门**

**JS三种事件模型：内联模型、脚本模型、****DOM2模型**

* 1. **事件处理函数**

**JS处理事件类型：鼠标事件、键盘事件、HTML事件**



* + 1. **鼠标事件**
       1. **click：**单击
       2. **dblclick：**双击
       3. **mousedown：**按下鼠标还未弹起
       4. **mouseup：**释放鼠标
       5. **mouseover：**移动到元素上方
       6. **mouseout：**移出
       7. **mousemove：**在元素上移动时
    2. **键盘事件**
       1. **keydown：**按下，若按住不放，会重复触发

window.onkeydown = function(){

console.log(111)

}

* + - 1. **keypress：**按下，若不松手，会重复触发
      2. **keyup：**释放键盘
    1. **HTML事件**
       1. **load：**
          1. 当页面完全**加载**后在window上面触发，或当框架集**加载**完毕后在框架集上触发
       2. **unload：**
          1. 当页面完全**卸载**后在window上面触发，或当框架集**卸载**后在框架集上触发
       3. **select：**
          1. 当用户选择文本框中的一个或多个字符触发
       4. **change：**
          1. 当文本框(text和textarea)内容改变且失去焦点后触发
       5. **focus：**
          1. 当页面或元素获得焦点时触发
       6. **blur：**
          1. 当页面或元素失去焦点是在window及相关元素上触发
       7. **submit：**
          1. 当用户点击提交按钮在<form>元素上触发
       8. **reset：**
          1. 当用户点击重置按钮在<form>元素上触发
       9. **resize：**
          1. 当窗口或框架的大小变化时在window或框架上触发
       10. **scroll：**
           1. 当用户滚动滚动条的元素时触发