复习：

Regular Expression, 规则的表达式，正则表达式

定义正则表达式对象：

var regexp1 = /\d{2,5}/igm ;

var regexp2 = new RegExp(‘\\d{2,5}’, ‘igm’);

使用正则表达式对象：

regexp.xxx( string ) ; //test() exec() complile()

string.xxx( regexp ); //replace() match() split() search()

正则表达式的写法：

/abc/

/ab{2,}/

/ab+c\*d?/

/\d\D\w\W\s\S./

/[a-z\*]/

/^a\bcc\Ba$/

/\d(?=ab)/

/\d(?!ab)/

/aa(bb)(cc)\1\1\2/ RegExp.$1

1.String对象的方法中可以使用RegExp作参数的

str.replace( regexp, replacement );

str.match( regexp ) 返回字符串中所有匹配正则表达式的子串

str.split( regexp )

str.search( regexp ) 作用于indexOf()类似，但参数支持正则表达式

2.贪婪匹配(Greey Match和懒惰匹配(Lazy Match)——面试题

**贪婪匹配：**\* + {x,y} 默认情况下，频次元字符会“尽可能多”的匹配

**懒惰匹配：**在频次元字符后面加?，转换为懒惰匹配：会“尽可能少”的匹配内容

复习：

JS中的数据类型：

1. 值类型
2. 特殊类型
3. 引用类型 Object
   1. 原生对象：ECMAScript标准定义

Array String RegExp Number Boolean Math Date Error

* 1. 宿主对象：window、document
  2. 自定义对象：

3.Math对象

static : 静态的，JS中的保留字

非静态属性： var s1 = new String(‘abc’); s1.length

非静态方法： var patt = new RegExp(‘’,’’); patt.test(‘’);

总结：非静态成员，使用时必须先构建一个对象，使用对象的引用来调用

静态属性： RegExp.$1 Math.PI

静态方法： Math.abs(-5)

总结：静态成员，使用时无需构建对象，直接使用类型名来调用

Math类型中的所有属性和方法都是静态的，故无需创建Math对象

Math类型常用属性和方法：

Math.PI

Math.round()

Math.ceil()

Math.floor()

Math.random()

练习：

(1)生成count个min-max之间的随机数，保存在一个数组中

(2)从Tom、Mary、John这三个姓名中随机取出一个姓名

var arr = [];

var i = parseInt(Math.random()\*arr.length)

arr[ i ]

4.Boolean类型

Boolean类型是（boolean值类型）对应的引用类型

var b1 = true; //值类型变量 typeof b1 => boolean

var b2 = new Boolean(true); //引用类型对象 typeof b2 => object

5.Number类型

Number类型是（number值类型）对应的引用类型

var n1 = 123; //值类型

var n2 = new Number(123); //引用类型对象

var n3 = new Number(‘123.45’); //

var n4 = new Number(‘123.45ab’); //

面试题：new Number(string) 和 parseInt(string)/parseFloat(string) 有和区别？

面试题：

不使用parseInt()方法，自己创建一个myParseInt(string)把一个string解析为整型值。

Number类型常用静态属性：

Number.NaN

Number.MAX\_VALUE

Number.MIN\_VALUE

Number.POSITIVE\_INFINITY

Number对象的常用方法：

n1.toExponential() 转换为指数法表示形式

n1.toFixed(x) 四舍五入到指定小数位数的数组

n1.toPrecision(x) 转换为指数法表示形式，具有指定的有效位

n1.toString(2/8/16) 转换为指定进制的字符串表示

练习：

(1)将12345.6789以指数形式输出、保留两位小数形式输出、二进制/八进制/十六进制形式输出

(2)不使用parseInt()方法，自己创建一个myParseInt(string)把一个string解析为整型值。

5.Date类型

表示日期和时间(到毫秒精度) 1s=1000ms

计算机中以一个长整型的数字来表示时间，值指距离“计算机元年“经过了多少毫秒。

var n1 = new Date(1000\*3600);

var t = n1.getTime(); //3600000

Date常用方法：

d1.getYear() 返回存储的年份-1900

d1.getFullYear() 返回存储的年份

d1.getMonth() 返回存储的月份 0~11

d1.getDate()

d1.getHours()

d1.getMinutes()

d1.getSecond()

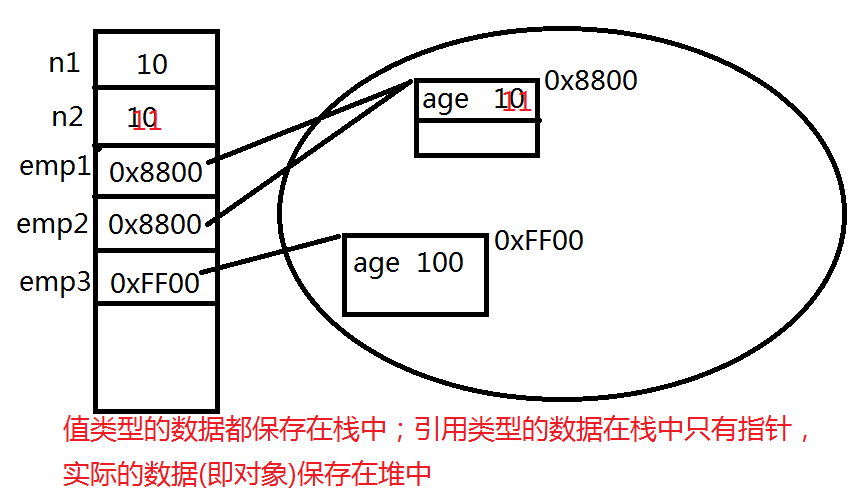
d1.getMilliseconds()

d1.getTime() 返回存储的时间距离计算机元年多少毫秒

d1.toLocaleDateString()

d1.toLocaleTimeString()

d1.toString()



6.错误处理

可抛出(throw)的对象：其它语言分为“异常”和“错误”，JavaScript中能够抛出的对象全部称为Error——可能出现、也可能不出现的问题。遇到错误，JavaScript解释器会创建一个Error对象，其中有name和message属性，保存着错误的名称/提示消息；并把此对象抛出来给用户。用户若处理了（有的解释器还提供了一个stack属性，提供了函数调用的追踪信息），则程序还可以继续执行；若程序没有处理，程序会终止执行。JavaScript中提供的错误类型：

**Error**  所有Error共同父类，但解释器只可能抛出某种具体的错误对象，而不会直接抛出一个Error对象

**SyntaxError**  语法错误

**TypeError** 类型错误

**ReferenceError** 引用错误

**EvalError**  非法调用eval()函数抛出的错误，很少见

**RangeError** 范围错误，如new Array(-1)

**URIError**  URI错误，encodeURI() / decodeURI() / encodeURIComponent() / decodeURIComponent()若发现了非法的URI可能抛出URIError

总结：ECMAScript中定义的所有原生的引用类型

Number Boolean String

Array RegExp Math Date

Error SyntaxError ReferenceError TypeError RangeError EvalError URIError

Object Function

Global (其存在的目的是为了搜集js中全局函数，如parseInt())

课后练习：

