正课:

1. RegExp

2. Math

3. \*\*Date

1. RegExp:

什么是: 专门封装一条正则表达式，并提供用正则表达式执行验证和查找的API

何时: 2种:

1. 验证字符串的格式

2. 即查找内容，又查找位置

如何:

创建: 2种:

1. 直接量: var reg=/正则/ig;

何时: 如果正则表达式是固定不变的

问题: //之中不支持js表达式,也就是不支持动态生成正则表达式

2. 用new: var reg=new RegExp("正则","ig")

何时: 如果正则表达式需要动态拼接生成

API:

1. 验证字符串的格式:

var bool=reg.test(str);验证str是否符合reg的规则要求

返回: bool，可直接当做判断条件

问题: test默认只要部分匹配就返回true

解决: 从头到尾完整匹配: 必须前加^后加$

2. 即查找所有关键词内容，又查找位置:

reg.exec(str) 依次查找str中下一个关键词的内容和位置

返回值: 数组: [0: 关键词内容, index: 下标位置]

如果找不到, 返回null

问题: 每次只返回本次找到的一个关键词

解决: 用循环反复调用reg.exec

原理: 每次调用exec，做3件事:

1. 将本次找到的关键词保存在数组0位置

2. 将本次找到的关键词位置保存在数组index位置

3. 修改reg对象的lastIndex=index+关键词的字数

reg.lastIndex : 记录下次开始查找位置, 默认为0

2. Math:

什么是: 专门保存数学计算的常量和函数的对象

何时: 只要进行数学计算

如何:

创建: Math不能new！不用创建！

所有常量和API都用Math直接调用！

API:

取整:

1. 上取整: 只要超过，就取下一个整数

Math.ceil(num)

2. 下取整: 舍弃小数部分

Math.floor(num) 只能对纯数字

vs parseInt(str) 能去掉数字之后非数字字符，再取整

3. 四舍五入取整: 够5进1，不够舍弃

Math.round(num) 优: 返回number类型的数值,可直接用于计算。

缺: 不能指定小数位数，只能取整

vs n.toFixed(d) 缺: 返回字符串类型，计算前必须先转换为数字。

优: 可指定小数位数，按任意小数位数四舍五入。

自定义:

乘方和开平方:

乘方: Math.pow(底数,幂)

比如: 10的2次方: Math.pow(10,2)=100

开平方: Math.sqrt(num)

比如: Math.sqrt(9) = 3

最大值/最小值:

Math.max(值1,值2,...)

Math.min(值1,值2,...)

问题: 不支持获得数组中的最大值

解决: Math.max.apply(null,arr)

其中: apply表示"调用"的意思

还可自动打散数组类型参数！

随机数:

0<=Math.random()<1

公式: 在任意范围min~max内生成随机整数:

var r=parseInt(Math.random()\*(max-min+1)+min);

简化: 如果从0开始:

var r=parseInt(Math.random()\*(max +1));

3. \*\*Date:

什么是: 专门封装一个时间点，并提供操作时间的API

何时: 只要存储时间或计算时间

如何:

创建: 4种:

1. 创建日期对象并自动获得当前系统时间:

var now=new Date();

强调: new Date()只能获得客户端本地时间

2. 创建日期对象并封装自定义时间:

var date=new Date("yyyy/MM/dd hh:mm:ss");

new Date(yyyy,MM-1,dd,hh,mm,ss)

3. 复制一个日期对象:

为什么: 日期计算都是直接修改原日期对象

计算后，无法保存计算前的原日期

何时: 如果用户希望同时保留计算前后的开始和结束时间时，都要先将开始时间复制一个副本，再用副本计算结束时间。

如何: var date2=new Date(date1);

4. 用毫秒数创建日期对象:

为什么: 将来数据库中保存的时间都是毫秒数

何时: 只要将毫秒数转化为当地时间格式:

var date=new Date(ms)

其中new Date会将ms转化为操作系统当前所在时区的对应时间显示

获得日期中的毫秒数: var ms=date.getTime();

原理: 其实日期对象中保存的是1970年1月1日0点至今的毫秒数

为什么: 毫秒数不受时区影响

API: 3句话

1. 8个单位:

FullYear Month Date Day ——没有s

Hours Minutes Seconds Milliseconds 都以s结尾

2. 每个单位都有一对儿get和set方法:

其中: get负责获得指定单位上的数值

set 负责修改指定单位上的数值

比如: date.getFullYear() date.getMonth()...

date.setFullYear(n) date.setMonth(n)...

强调: setXXX(n)可自动调整时间进制:

特例: Day没有set

3. 取值范围:

除date外，其余都是从0开始到进制-1结束

date从1开始，到31结束

Month: 0~11 都比现实中小1

Date: 1~31 和现实相同

Day: 0~6和现实相同 星期日为0

Hours: 0~23和现实相同

Minutes/Seconds: 0~59和现实相同

计算:

1. 两日期对象可相减: 结果毫秒差

何时: 倒计时

2. 对每个单位做加减:

3步:

1. 取值: var d=now.getDate(); //d=8

2. 做加减: d+=30; //d=38

3. 放回去: now.setDate(d) //自动调整进制

// 6/7

简化: now.setDate(now.getDate()+n)

日期格式化:

.toString() 返回当地时间格式的完整版本

.toLocaleString() 返回当地时间格式的简化版本

.toLocaleDateString() 仅保留日期部分

.toLocaleTimeString() 仅保留时间部分

.toGMTString() 获得0时区的标准时间