正课:

1. Date

2. Error

3. \*\*\*Function

1. Date:

什么是: 封装一个时间，并提供操作时间的API

何时: 只要在程序中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: 只能获得客户端系统时间

2. 创建日期对象并封装自定义时间

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

3. 将毫秒数转化为日期对象:

日期对象本质: 日期对象中实际保存的是1970年1月1日至今的毫秒数

为什么: 用文字表示时间的两个问题:

1. 字符串不便于计算

2. 会受时区影响

用毫秒数的优点:

1. 便于计算

2. 不会受时区的影响

毫秒数的问题: 人看不懂，且没有API

解决: 用new Date()将毫秒数转为日期对象，并使用日期对象提供的简化API操作时间

如何: var date=new Date(ms);

4. 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

何时: 如果希望同时保留计算前后的新旧两个时间时

如何: var date2=new Date(date1)

API: 3句话:

1. 8个单位:

FullYear Month Date Day

Hours Minutes Seconds Milliseconds

2. 每个单位上都有一对儿get/set方法

其中: date.getXXX() 获取一个单位上的数值

date.setXXX(n) 修改一个单位的数值

特例: Day 没有set

3. 取值范围:

Month 0~11 计算机中的月份值比现实中小1

Date 1~31

Day 0~6

Hours 0~23

Minutes/Seconds 0~59

计算: 2种:

1. 两日期对象可相减: 结果是毫秒差

2. 对任意单位做加减:

3步:

1. 取值: var d=date.getDate();

2. 计算: d+=7

3. 改回去: date.setDate(d);

简写: date.setDate(date.getDate()+7);

福利: setXXX() 可自动调整时间进制

问题: setXXX() 直接修改原日期对象

解决: 先复制日期对象副本，再用副本计算

日期格式化:

date.toString() 转为当地时间的完整格式

date.toLocaleString() 转为当地时间的简化版格式

date.toLocaleDateString() 仅保留日期部分

date.toLocaleTimeString() 仅保留时间部分

date.toGMTString() 转为0时区国际标准时间

2. Error:

什么是: 当错误发生时，自动创建的，封装错误信息的对象

什么是错误: 程序执行过程中，导致程序无法继续执行的异常情况

什么是错误处理: 即使程序出错，也不会中断退出的一种机制

为什么: 程序只要出错，都会闪退

何时: 只要希望，即使程序出错，也不会中断退出

如何:

try{

可能出错的正确代码

}catch(err){//只有在出错时，才触发

//错误处理代码

//提示错误信息，保存进度，记录日志

}

问题: 会额外创建错误对象，降低效率和浪费内存

解决: 用if else提前预防错误的发生

主动抛出错误:

throw new Error("错误提示")

3. Function

什么是: 专门封装一段代码段的对象

何时: 只要重用一段代码时

如何:

创建: 3种:

1. 用声明方式:

function 函数名(参数列表){

//函数体

//返回值

}

问题: 声明提前(hoist) 破坏程序执行的顺序

解决:

2. 用赋值方式创建函数:

var 函数名=function(){ ... }

优点: 避免声明提前

3. 用new:

var 函数名=new Function("参数","参数",...,"函数体")

重载overload:

什么是: 相同函数名不同参数列表的多个函数，在调用时，可根据传入参数的不同，自动选择匹配的函数执行

何时: 只要一个函数根据不同的参数，执行不同的任务时。

为什么: 减少API的数量，减轻调用者的负担

如何:

问题: javascript默认不支持重载

因为: 不允许多个同名函数同时存在

解决: arguments

什么是arguments: 函数内自动创建的，收集所有传入函数参数值的 类数组对象

vs 数组: 相同: 1. 下标, 2. lenght, 3. 遍历

不同: 类型不同， API不通用

何时: 只要不确定传入函数的参数个数时

匿名函数:

什么是: 函数创建后，不保存到任何变量的函数

何时: 如果一个函数只用一次时

为什么: 节约内存——匿名函数调用后自动释放

如何: 2种:

1. 回调: 创建函数后，自己不调用，而是给另一个函数去调用:

3个:

arr.sort(function(a,b){return a-b})

str.replace(/正则/ig,function(kw){ return 新值 })

setInterval(function(){ ... },1000)

2, 自调: 函数定义后，立刻调用自己

问题: 禁止使用全局变量

为什么: 全局污染，内存泄漏

解决: 用匿名函数自调包裹所有变量和操作

闭包closure:

什么是: 即重用变量，又保护变量不被篡改的机制

为什么: 全局变量和局部变量都有不可兼得的优缺点:

全局变量: 优: 随处可用，可重复使用

缺: 造成全局污染和内存泄漏

局部变量: 优: 仅函数内可用

缺: 不可重用

何时: 如果希望即重用一个变量，又保护变量不被篡改时

如何: 3步:

1. 用外层函数，包裹内层函数和受保护的变量

2. 外层函数返回内层函数到外部

3. 调用外层函数，获得内层函数，保存在变量中