正课:

1. Date

2. Error

3. \*\*\*\*Function

创建，重载，匿名函数，

\*\*\*\*作用域和作用域链

\*\*\*\*\*闭包

1. Date:

什么是: 封装日期和时间, 提供操作日期和时间的API

何时: 只要存储日期和时间，都用Date类型的对象

如何:

创建: 4种:

1. 获得当前系统时间: var now=new Date();

强调: 只能获得客户端当前操作系统的时间

2. 保存自定义时间:

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

3. 复制一个日期对象:

为什么: 今后，日期对象的计算，都直接修改原日期对象。计算后，原日期无法保留。

何时: 只要用户希望同时保留计算前后的开始和结束时间时，都要先复制副本。再用副本计算结束时间。

如何: var date2=new Date(date1);

4. 将毫秒数转化为当地时间:

为什么: 系统中存储时间，都是使用毫秒数

何时: 只要将毫秒数转为当地时间显示

如何: var date=new Date(ms);

补: 如何获取毫秒数: var ms=date.getTime();

原理: 日期对象中，保存的是1970年1月1日0点0分0秒至今的毫秒数

为什么: 文字会受时区的影响，而毫秒数不受时区的影响

API:

1. 8个单位:

FullYear Month Date Day 没有s结尾

Hours Minutes Seconds Milliseconds 都有s结尾

2. 每个单位都有一对儿getXXX/setXXX方法

其中: getXXX() 负责获取指定单位的数值

setXXX() 负责修改指定单位的数值

特例: Day没有set方法

3. 取值范围:

Month: 0~11 计算机中每个月份都比现实少1

读取时，+1修正; 修改时，-1修正

Date: 1~31 Day: 0~6

Hours: 0~23 Minutes/Seconds: 0~59

日期计算: 2种:

1. 两日期对象可相减，结果是毫秒差

2. 对任意单位的数值做加减: 3步

1. 取出单位的值: date.getXXX()

2. 做加减

3. 设置回去: date.setXXX(n)

强调: 1. setXXX()可自动根据n，调整时间进制

2. 如果希望同时保留计算前后的新旧时间，则必须先复制副本，再用副本计算。

其实可简写:

date.setXXX(date.getXXX()+n)

日期转字符串:

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() 转为当地时间的简化版格式，仅保留日期部分

date.toLocaleTimeString() 转为当地时间的简化版格式，仅保留时间部分

date.toGMTString() 转为国际标准时间(0时区)

自定义:

Error

什么是错误: 程序运行过程中，导致程序无法正常执行的问题。

程序发生错误后，都会强行中断退出

什么是错误处理: 当程序发生错误时，依然保证程序不中断退出的机制

何时: 只要有可能发生错误的代码，都要提前使用错误处理

如何:

try{

可能出错的正常逻辑代码

}catch(err){//只有出错才执行，不出错不执行！

一旦出错，才执行的错误处理代码:

提示用户, 记录日志, 保存数据

}

其中: err: 错误对象: 专门封装错误信息的对象

发生错误时，自动创建！

主动抛出错误: throw new Error("错误信息")

何时: 函数的作者向函数的调用者抛出错误，提示调用者错误的使用了函数。

鄙视: 错误的类型: 6种

SyntaxError: 语法错误

ReferenceError: 引用错误: 要用的变量/对象，没找到！

TypeError: 类型错误

错误的使用了类型: 1. 错误的使用了()或[]

2. 错误的调用了对象的方法

RangeError: 范围错误: 参数超范围

EvalError; URIError;

Function

创建: 3种:

1. 声明: function 函数名(参数列表){

函数体;

return 返回值

}

问题: 被声明提前

解决:

2. 直接量: var函数名=function (参数列表){...}

优: 不会被声明提前!

揭示: 函数其实也是一个引用类型的对象

function是创建一个新函数的意思

函数名其实就是一个引用函数对象的简单变量

3. 用new:

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

鄙视: 以下创建函数正确的是:

function cmp(a,b){return a-b;}√

var cmp=function(a,b){return a-b;};√

var cmp=new Function(a,b,"return a-b"); X

var cmp=new Function("a","b","return a-b"); √

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，自动选择对应函数执行。

为什么: 减少函数的个数，减轻调用者的负担

何时: 只要一项任务，根据不同的参数，有不同实现逻辑时

问题: js语法默认不支持重载！

原因: js中不允许多个同名函数同时存在！

解决: 用arguments对象

什么是: 函数中，自动创建的

自动接收所有传入函数的参数值的

类数组对象(object like array)

相同: 1. 下标, 2. .length, 3. for遍历

不同: 类型不同! API不通用！

何时: 只要函数的参数个数不确定时，都可用arguments接住所有参数值

问题: 今后，还需不需要参数变量！

参数的作用:提醒调用者，如何正确使用函数！

一般自定义的参数名，都是见名知意，且都非常简单

所以，函数都要优先使用参数变量接收参数值