正课:

1. Date

2. Error

3. \*\*\*\*Function

创建; 重载; 匿名函数;

1. Date

什么是: 保存一个日期或时间，提供操作日期和时间的API

何时: 只要存储或操作日期和时间

如何: （PHP中:日期与时间相关扩展->日期和时间->Date/Time函数）

创建: 4种:

1. 获得系统当前日期和时间:

var now=new Date();

强调: 只能获得客户端本地时间，无法获得服务器时间

2. 自定义日期和时间:

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

问题1: 不同编程语言中存储日期的格式不一致!

问题2: 字符串不便于计算和大小比较

问题3: 文字会受时区影响

解决: 不用字，也不用数组/对象来表示时间

用毫秒数

Date对象的本质: 其中保存了1970年1月1日至今的毫秒数

为什么: 1. 便于计算和比较大小

2. 数字在所有编程语言中都统一支持

3. 数字不受时区影响: 因为数字是一个时间段

3. 将ms转化为客户端日期对象:

var date=new Date(ms);

\*\*\*总结: 今后，在网络间传输时间或在服务器存储时间，都用整数（毫秒数/秒数）

4. 复制一个日期对象:

为什么复制: 日期计算默认都直接修改原日期对象

何时复制: 如果希望同时保留计算前后的开始时间和结束时间时，就要先复制开始时间，再用副本计算结束时间

如何: var date2=new Date(date1);

API: 3句话

1. 单位: FullYear Month Date Day ——都没有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()负责获取单位的数值

比如: now.getFullYear() 2017

now.getHours() 11

setXXX(n) 负责修改指定单位的数值

比如: now.setFullYear(2018)

now.setDate(30)

福利: setXXX(n)

其中:n可以是任意值!

setXXX(n)可自动调整时间进制!

特例: day没有setDay()! 因为day是推算出来的

3. 取值范围:

Month: 0~11

强调: js中的月份比现实中小1

读取或修改时，可能需要+1或-1修正

Date: 1~31

Day: 0~6 日~六

Hours: 0~23

Minutes/Seconds: 0~59

日期计算: 2种情况:

1. 两日期对象可直接相减: 结果是毫秒差!

2. 对任意单位做加减:

3步:

1. 先取值: var d=now.getDate();

2. 做加减: d+=30;

3. 放回去: now.setDate(d); //setXXX()可自动调整时间进制

问题: setXXX() 直接修改原日期对象

解决: 先复制副本，再用副本计算

简写: now.setDate(now.getDate()+30);

将日期转为字符串:

date.toString(); 转化为当地时间的完整格式

date.toLocaleStirng(); 转化为当地时间的简化版格式

date.toLocaleDateString(); 简化版格式，仅保留日期部分

date.toLocaleTimeString(); 简化版格式，仅保留时间部分

date.toGMTString(); 转化为国际标准时间

自定义格式化日期函数

2. Error:

什么是错误(bug): 程序执行过程中，导致程序无法正常执行的异常情况。

错误处理: 即使程序出错，也保证程序不会强行退出的机制

为什么: 只要程序出错，都会强行中断退出

何时: 只要希望程序即使出错，也不中断退出

如何:

try{

可能出错的正确代码

}catch(err){//err: 错误对象

只有发生错误时，才执行的错误处理代码:

3件事: 1. 提示用户, 2. 保存进度, 3. 记录日志

}

错误对象: 发生错误时，自动创建的，保存错误信息的对象

问题: 放在try catch中的代码，执行效率可能降低，且生成错误对象，占用内存空间

解决: 如果可以提前预知错误，则首选if...else...来提前预防错误。

错误类型: 6种

SyntaxError: 语法错误

ReferenceError: 引用错误, 要用的变量没找到

TypeError: 类型错误: 错误的使用了对象的方法或()或[]

RangeError: 范围错误, 参数超范围

URIError: URI错误, 如果请求所使用的url拼接错误

EvalError: Eval错误

抛出错误:

何时: 函数的作者向函数的调用者抛出自定义错误对象，告知调用者错误的使用了函数。

如何: throw new Error("自定义错误提示信息")

3. \*\*\*\*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: 会被声明提前！

解决:

2. 赋值:

var函数名=function (参数列表){ 函数体; return 返回值}

优势: 不会被声明提前

揭示: 函数名其实仅是一个普通变量

函数其实是一个保存代码段的引用类型对象

函数名通过地址引用函数对象

3. 用new:

var 函数名=new Function("参数1","参数2",...,"函数体;...")

强烈不推荐使用:

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");

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，自动选择对应的函数执行

为什么: 减少API的数量，减轻调用者的负担

何时: 一件事，只是根据参数的不同，执行不同的逻辑时

如何: js语法默认不支持重载

为什么: js不允许多个同名函数同时存储在，最后创建的会覆盖之前创建的所有

变通: arguments

arguments:

什么是: 函数调用时自动创建的

接收所有传入函数的参数值的

类数组对象

类数组对象: 长的像数组的对象

vs 数组: 相同: 1. 下标, 2. .length, 3. for遍历

不同: 类型不同! API不通用!

何时: 只要传入函数的参数不确定时

是否还需要参数? 需要：

1. 参数名简单易用，且见名知义

2. 参数可以指导调用者正确使用函数

匿名函数:

什么是: 创建函数时，不指定函数名的函数

为什么:

1. 节约内存

2. 划分临时作用域

何时: 只要一个函数使用后，希望立刻释放时

如何:

1. 回调(callback): 将一个函数，交给另一个函数去自动调用。

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

xhr.onreadystatechange=function(resText){}

btn.onclick=function(){}

str.replace(/正则/,function(kw){return xxx});

2. 自调: 定义函数后，立刻调用自己

何时: 几乎所有自定义的脚本，都要放在匿名函数中

为什么: 避免使用全局变量，避免全局污染

如何:

1. (function(...){...})()

2. +function(...){...}()