正课:

1. Error

2. \*\*\*Function

创建

重载

匿名函数

\*\*\*\*作用域和作用域链

\*\*\*\*\*闭包

1. Error

什么是Error: 程序执行过程中，导致程序无法继续正常执行的情况

一旦发生错误, 程序立刻中断退出

什么是错误处理: 即使程序出错，也保证程序不会中断退出的机制。

为什么: 为了保证程序即使出错，也不退出!

何时: 只要程序中，可能发生错误的地方，都要进行错误处理。

如何:

try{

可能出错的代码

}catch(err){//只在发生错误时自动执行

错误处理代码: 1. 提示用户, 2. 保存进度, 3. 记录日志

}

err: 错误对象: 在错误发生时，自动创建的，保存错误信息的对象

性能: 放入try catch中的代码，即使不出错，也会降低效率。

解决: 绝大多数try catch都可提前预知，并用if...else代替.

主动抛出错误:

何时: 在团队协作开发中，程序的作者，向程序的调用者，抛出错误，用来告诉调用者，错误的使用了程序。

如何: throw new Error("错误提示")

错误的类型: 6种:

SyntaxError: 语法错误

ReferenceError: 引用错误, 要用的变量/函数没找到！

RangeError: 范围错误, 参数超范围

TypeError: 类型错误, 错误的使用类型调用函数或访问元素:

EvalError; URIError;

2. \*\*\*Function:

创建: 3种:

1. 声明方式:

function 函数名(参数列表){ 函数体; return 返回值}

问题: 会被声明提前

2. 直接量:

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

特点: 不会被声明提前

揭示: 函数其实是一个引用类型的对象，

函数名仅是一个普通的变量而已

函数名中保存着函数对象的地址，以此引用函数

3. new :

var 函数名=

new Function("参数1","参数2",...,"函数体;return 返回值")

何时: 如果函数体或参数列表需要动态生成

问题: 效率低

鄙视: function fun(a,b){return a-b;}

var fun=function(a,b){return a-b;}

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

var fun=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，自动调用对应函数执行.

为什么: 减少API的数量，减轻调用者的负担

何时: 只要一项任务，因为传入参数的不同，需要执行不同的操作时。

如何:

问题: js语法不支持重载！

因为: js中不允许多个同名函数同时存在！

解决:

1. 只定义一个函数，包含所有操作流程, 不要定义任何参数！

2. 在函数内用arguments接收所有参数值

arguments: 每个函数中自动创建的

自动接收所有传入函数参数值的

类数组对象

何时: 重载时，不确定传入函数的参数个数，就可用arguments接住所有传入函数的参数

匿名函数:

什么是: 创建函数时，不定义函数名称

何时:

1. 临时使用一个函数，使用后自动释放，节约内存

2. 划分临时作用域,避免全局污染

如何:

1. 回调:

2. 自调: 定义匿名函数后，立刻调用自己。调用一次后，立刻释放！

如何: 1. (function(){....})()

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

总结: 今后，所有代码，都应写在匿名函数中!

\*\*\*\*作用域(scope)和作用域链(scope chain)

什么是作用域: 一个变量的可用范围

专门保存变量的对象

2种:

全局作用域: window

函数作用域: AO

函数的生命周期:

1. 程序开始执行前

创建ECS(执行环境栈): 保存引擎要执行的函数

浏览器启动时，会优先调用浏览器的主程序main，压入ECS

主程序会创建全局作用域对象window

2. 定义函数时

在window中定义函数名变量

创建函数对象，保存函数的内容

函数对象的scope属性，指回函数来自的作用域对象

函数名变量通过地址引用函数对象

3. 调用函数时

在ECS中添加当前函数的调用

创建函数作用域对象, 保存所有局部变量

设置函数作用域对象的parent指向函数来自的作用域对象

控制了变量的使用顺序:

先用函数作用域对象中的局部变量

局部没有，才去全局找

4. 函数调用后:

ECS中当前函数的调用出栈!

导致: 函数作用域对象无人使用，被释放

导致: 局部变量一同释放！

什么是作用域链: 由各级作用域，逐级引用形成的链式结构

2个作用:

1. 存储了所有的变量

2. 控制着变量的使用顺序

\*\*\*\*\*闭包:

什么是: 即重用一个变量，又保护变量不被污染的一种机制

为什么:

全局变量: 优: 随处可用，可重复使用

缺: 易被篡改。易造成全局污染

局部变量: 优: 不会被篡改, 不会造成全局污染

缺: 不可重用！

何时: 即重用变量，又保护变量不被污染

如何: 3步:

1. 用外层函数包裹受保护的变量和操作变量的内层函数

2. 外层函数将内层函数返回到外部

3. 使用者调用外层函数，获得内层函数

鄙视: 闭包形成的原因:

外层函数的函数作用域对象无法释放

导致: 外层函数的局部变量被保存下来——可重用

鄙视: 画简图:

1. 受保护的变量: 都是外层函数的局部变量

2. 操作变量的内层函数: 只有内层函数才有资格访问受保护的变量