类型检查 typeof 系统原生的引用类型 自定义引用类型无法获取名字？funciton.name?

.constructor.name?

关于undefined 未初始化 未声明

和null 类型是object

For in 遍历属性不能是null undefined 返回属性名

in

数值范围 数量超出范围得到无穷大Infinity isFinite() NaN isNaN()---会发生隐式转换

非数转数 parInt parseFloat

转字符串 toString 基础值 对象值转字符串？ Null undefine 没有

原型 继承 变量作用域链 内存模型 闭包 高阶函数

函数对象

一个对象的隐藏属性和方法 constructor hasOwnProperty(非原型属性，即实例属性)

isPrototypeOf(obj) 是否其他对象的原型 valueOf()

2、操作符会发生隐式转换 == >= = +-

一元加 = -a 表示正负 字符串,bool会自动转数字. ++递增也会强制转换

逻辑运算符会发生隐式转换 ！非 对Null 对undefined NaN 0 “”

与&&会短路 返回值不一定是bool 可以是对象，数，undefined,null等

**请不要在与或里面写非布尔值**

乘法超出范围得到无穷大 无穷大的计算也是无穷大 会隐式转换非数

加法 -0 +0

计算NaN得到NaN

计算对象会用valueOf转换 没有就toString转换

字符串拼接 “”+””

比较：字符串会比较编码 字符和数比较，会转为数字或者NaN 对象会转valueOf

比较时都会转数值或者NaN

相等：会发生转换为数或NaN

逗号操作符——同时进行

3、参数 可以传任意个参数 定义时可以使用arguments对象接受隐式参数 arguments.length 不能重载

四、变量作用域

1、引用类型的对象可以动态增加属性

2、复制赋值 基本类型复制的是值 引用类型复制的是引用

3、参数属于局部变量 arguments[]数组中的一个值 ，调用函数赋参，就是给这个局部变量赋值的过程 arguments[0] = obj; 在内部对参数进行赋值操作是影响不了外面的，但是可以借用对象参数，来修改被引用对象的属性

4、检查引用类型 Instanceof

5、执行环境 context 每个环境一个变量对象 ，保存所有函数和变量 一个函数一个执行环境

作用域链，连接自己和外层的变量对象，一层层直到全局 会从最底层识别同名标识符，找不到的名字就往外层找

尽量不要多层嵌套定义函数（有时候需要闭包，比如回调函数

**在函数内部不加var，会被定义到全局，容易导致很多bug,比如每次执行函数值未重置**

6、延长作用域链 try catch

7、垃圾回收

局部变量 在堆和栈上分配

标记清除：声明时（进入环境），打标记 离开时打离开标记

引用计数

**优化方式：解除引用 设置为null——用于全局对象和全局对象的属性**

五、引用类型

1、对象

对象字面量和new Object(), {a:1,b:””,c:[]} 无属性对象{} （其实有原型属性和方法） 对象作为参数也不错，可以传递多个值

属性名可以是字符串 “property”:[], 也**可以是数值**，2：“ok”会自动转字符串

**对象可以当成关联数组,hash表使用**

2.数组

New Array(3) 数组字面量 数组长度.length可赋值，可以添加新项或者删除

**数组没初始化默认元素值是undefied**

Array.isArray()

数组转字符串 toString() ，用逗号分隔 如何想转为不用逗号分隔的字符串，用.join()

valueOf 仍返回数组

push pop shift unshift(在前方增加项) 会直接改变原数组

重排序： reverse sort 比较函数 compare(value1,value2){return} 会返回新数组 连接:.concat(array,”value”) slice(1,4)子数组，不影响原来

Splice() 影响原数组，增删改 返回被删除的数组

位置方法：indexOf() 查找元素位置，没找到返回-1

迭代方法：把数组的各元素传入某个函数参数（多用于比较），是否全部满足every() 至少满足一项 some() 筛选出满足的元素filter() (item)=>{reutrn item>0}

利用值来执行函数forEach() foreach通常用来利用数组值修改外部 获得返回值数组（映射）map() map通常将数组进行某种变换，获取返回数组

3.日期 解析字符串Date.parse() 直接new Date(string)也行 Date.UTC(y,m,d,h,m,s) 也能new Date(y,m,d,h,m,s)可以省略

或者现在时间 new Date() Date.now() 返回的是毫秒

toString toLocalString 返回格式 valueOf()返回的是毫秒

特殊格式字符串 toDateString toTimeString

Get set方法getTime setTime 日月年时分秒 星期 getFullYear getMonth

4、正则

使用字面语法 /pattern/flags g全局（匹配多项而不是一项） m多行，默认只匹配一行？ I无视大小写 .匹配任意字符除了空白

属性： g i m 起始位置 方法：exec(text) 返回匹配数组mathes 属性，index,input原串 mathes[0] [1]

一次exec()只会匹配到一项， 数组每个元素对应到一个捕获组 只有一个捕获组就只有一个元素

捕获组（） 可以方便地访问到各个括号里的子匹配项——必须要满足所有括号才成立

.Test(string)检查字符是否匹配，用于if条件检测

静态属性

向前查找 开始结束^$

5、函数对象 函数是function的实例 匿名函数 函数声明会被解释器提升

函数可以作为参数 或者返回值 exec(func) {func()}

Arguments对象 有个callee属性 函数有调用者属性func.caller

This对象 返回函数的执行环境，全局函数,匿名函数 setInterval(function()=>{})

参数个数 func.length

Prototype原型 对象， **原型不可枚举，for in无效**

Apply call() 修改this, 传入参数数组，或者是逗号参数 bind **相当把函数修改为其他对象的方法了**

ToString 返回函数代码

6、包装类型 基本类型调用方法时会自动生成为临时的包装对象，然后销毁 1.toString

Typeof 数字转字符串 **转固定长度的小数** toFixed 转指数toExponential()

字符串方法：按索引找字符或者字符编码 charAt() 拼接concat 用加号代替

提取子串 substring 可以是负值

位置方法 查找字符位置indexOf() 可以指定查找位置，用于多个值查找，一直向前

删除空格 大小写转换 模式匹配：匹配 替换match search replace(全局替换)

编码转字符串 fromCharCode

HTML方法：可以用字符串生成标签

7、全局对象 global window uri编码 encodeURI替换空格

encodeURIComponent 替换非字母数字（用于后面） decodeURI

字符串解析eval

全局属性：undefined NaN Infinity Object Array Function Boolean Error

8、Math对象 常数

方法

六、对象 属性类型 数据属性 属性描述对象 特性 Object.defineProperty()

可配置 configurable 可枚举 enumerable可写 writable值value 使用define时，默认值是false

访问器属性 有getter 和setter函数 控制取值和赋值——可以修改行为

DefineProperties(obj,{property:{writable:}}

获取描述对象getOwnPropertyDescriptor() .value .configurable

2、对象创建： new Object() 和 {}不方便

工厂模式 create函数 new Object() 添加属性 ，return obj

使用字符串参数来创建对象

构造函数： A(a,b,c) {this.a} new的行为：自动创建新对象，this指向该对象，执行构造函数，其中的thia.a=a会给新对象增加新属性，新对象成为返回值

.constructor instanceof

构造函数作为普通函数调用，this是指向全局的，属性会加在window上 ，如果.call(其他对象)， 就会加到其他对象上

**构造函数，创建了太多的方法实例对象（function），占用空间，方法一般是不需要改的，可以公用, 每次匿名函数都会创建一个全新的方法。。。如果指向外部的命名方法，就不会重复了**

原型对象，prototype prototype={} 里面的方法可以被

所有函数都有prototype A.prototype.constructor

被创建的对象有\_\_proto\_\_属性，指向A.prototype isPrototypeOf getPrototypeOf

重写方法，会屏蔽掉原型

删除实例属性 delete

检查属性属于原型还是实例 hasOwnProperty()

In对原型 和实例属性都有用 for in只能访问可枚举属性，不能访问原型 包括toString等 Object.keys()访问可枚举属性，包括无效值 访问实例属性，无论是否可枚举

getOwnPropertyNames()

原型指向其他对象 prototype={} 此时指向Object,constructor需要修复，但是此时constructor变为可枚举

原型的修改会影响所有的对象

原生对象的原型 Array.prototype.sort 可以动态修改原生对象

**原型一般是用来共享方法的，属性共享的话，一个修改就会导致所有对象修改**

2、构造和原型结合 构造this定义属性 原型 prototype定义方法，并且prototype.constructor指向构造

3、动态原型 在构造中根据条件判断，动态在原型上添加方法

4、寄生构造 工厂方法的变体 内容和工厂一样 通过new调用，最后有一个reutrn obj(instance无法确定，没有原型，不建议使用)

5、稳妥构造 是工厂方法，不用new，return obj 传入参数，然后在方法中使用，形成了闭包，除了调用方法无法访问该属性（仍然没有instance和原型）

7、继承 原型链 A.prototype=new B() a.\_\_proto\_\_== A.prototype ==new B==b b.\_\_proto\_\_==B.prototype a.\_\_proto\_\_.\_\_proto\_\_

顺着原型链可以使用一切链条上的方法，和属性

要修复prototype.constructor 所有对象都继承了Object的原型

Instanceof isProtorypeOf

因为new B()是单独的对象，可以任意增加删除方法，不会影响到上层的原型链

给原型添加方法，千万不要用对象字面量，这样会重建新对象

**原型继承不只会继承原型链，也会继承实例属性， prototype= new B ==b ,b.name也会被继承下来**

不能传参

8、借用构造 A（）{B().call(this)} 内部调用B的构造就会把属性创建在自己的对象上 可以传入参数 ——父类原型方法无法访问

9、组合继承 属性和方法分开继承 内部调用B（）继承属性 A.prototype = new B()

继承方法 修复constructor 10、原型式继承 临时构造函数，F.protorype = obj

Return new F() Object.create() 浅复制 11、寄生式继承 先克隆，再增强

12、寄生组合 先调用父类构造

利用复制设置原型自己的原型设置成父类原型的**克隆**

七、函数表达式 匿名函数var say 此时不存在函数提升 2、递归 calle 3、闭包

可以访问另一个函数作用域（一般是自己被创建的函数），内部函数 形成特殊的作用域链

闭包可能是返回的函数，或者访问， 被访问的是函数的局部变量

[scope] 活动对象初始化,this,arguments,参数

[scope] 会保存作用域链，包含的是各变量对象的引用。。。闭包局部活动对象不会销毁

1. 闭包保存的是整个变量对象 for(i) { a[i] = function(){return i;} }

所有的函数都是引用的同一个变量对象中的i

立即执行函数 function(num) {return function(){return num} }(i）

每次执行都是一个新的匿名函数，有着不同的局部变量 function(){

Var num=i; return function() {return num} }() 这样才比较合适

**参数就是局部变量**

1. 匿名函数的执行具有全局性，this通常指向window setInterval() 对象中如果用到匿名回调函数 ，需要访问本对象的属性时，this一般是访问不到的，需要apply(this)

3、var that = this return function()

4、块级作用域 5、私有变量 可以用闭包来创造 利用匿名的立即执行函数

6、模块模式 单例 私有变量 私有方法

返回一个公共对象，对象方法调用了内部定义的私有属性和私有方法

7、增强的模块模式 内部增强属性和方法， return obj

八、窗口window

窗口位置 screenLeft screenTop 相对屏幕 移动窗口（不一定允许）moveTo moveBy

窗口大小 包含边框 outerWidth 不要边框 innerWidth

Ie:客户区大小clientWidth 调整大小resizeTo 禁用

打开窗口.open

2、定时器 持续调用 延时调用

3、对话框

4、location 查询字符串 hash host hostname href pathname port protocol search

查询字符串分割为数组，解码 遍历名值，name value,保存到哈希表中 search[name] =value

5、修改url .href .assign() .location=”” 修改location的任何属性都会重新加载，并生成记录 .replace()不生成记录

十、DOM节点 DOM树的解析

节点成树 共12种 文档 元素 特性 文本 文档类型doctype 注释

Xml cdatasection 文档框架（是作为节点仓库的，可以任何子节点， 不能加入文档）

节点类型.nodeType 节点名.nodeName 节点值 .nodeValue 子节点.childNodes （NodeList对象，类数组，是动态的） childNodes[0] .item(1) .length 可以用数组.call来使用数组方法 父节点.parentNode 兄弟节点 .previouSibling .nextSibling 首尾节点没有左/右兄弟，null 空对象 第一个子节点 ，最后的子节点.firstChild .lasetChild 没有则为null .hasChildren() 是否有子节点.ownerDocument 文档

节点操作 增加appendChild 插入 替换 移除removeChild

克隆节点（可以深复制，节点树） false浅复制，只复制自己

各节点的属性 方法 dom一致性检测

2、DOM操作 动态脚本 <script> 动态样式 动态表格 节点列表

3、扩展 选择符api 元素遍历 上一个 下一个 第一个孩子 可以连续调用

关于CSS类 类列表classList 焦点管理 当前焦点元素doc.activeElement

HTML： readystate head 字符集属性 自定义元素属性

插入html 插入文本

滚动出现在视口 scrollOIIntoView

子 children 是否后代 节点关系compareDocumentPosition()

4、dom3 xml命名空间前缀 xmnls 节点包含命名空间属性

类型的变化

5、访问样式 style对象 属性length **cssText**

计算样式

操作样式表sheet 创建规则

6、元素大小 偏移 客户区大小 滚动大小

7、遍历 深度优先

节点遍历器

树漫游：

8、范围接口 选择一个 文档区域，节点范围

十三、 事件处理程序 onclick 是一个函数对象——默认应该是undefine 或者null，需要重写 emmit(event){} 会触发该函数，会依次触发所有监听者的处理函数，并传入event对象 可以对一个事件添加多个处理函数

监听者列表 addEventlisener,会把当前对象加入到列表，并注册回调函数的引用

实际上是委托调用和传参而已 emmit(event) {div.onclick(event); }

AddEventlisener可以在捕获阶段处理 remove() 不能移除匿名

Event的属性 type target（实际想传的目标） **currentTarget（事件触发时传递给的目标）** 阻止冒泡 阻止默认行为

**由于存在委托机制，所以target不一定等于currentTarget ,委托，获取现在触发事件的元素，然后通过id等属性来确定元素类型，进行不同的处理， 可以处理单个委托或者多个元素的委托**

Ui事件 加载load unload error abort select scroll

图像上指定load script new Image() .src .onload

Resize 滚动 scrollLeft scrollTop可以控制滚动

1. 焦点 blur focus 不冒泡 focusin focusout冒泡
2. 鼠标 点击双击 按下 松开 进入元素 离开元素 内部移动 进入另一个元素 从其他元素进入

客户区坐标 页面坐标 屏幕坐标 修改键状态

相关元素relatedTarget

按下的键 button 滚轮事件

触摸

5、键盘 文本输入 键码keyCode 字符编码charCode是keypress的属性

Input

1. dom变动事件 插入删除 7、右键菜单 阻止默认，弹出自己的菜单

8、准备状态变更 9、hashChange **url的#变更**

10、旋转 11、触摸 12、手势

**13、事件模拟： 创建事件对象 createEvent 触发 dispatchEvent(event)**

**传入各种参数值模拟鼠标和键盘事件 （网页无法控制鼠标，可以用dom模拟假鼠标，哈哈）隐藏原鼠标，cursor:none**

14、自定义事件 initCustomEvent()

十四、表单属性 elements 所有控件 action enctype method

1、控件属性 name form type 控件类型 value值 disabled readOnly .focus()

Onsubmit onreset blur() 自动聚焦

Change事件失焦且改变时 select选项改变时

Input属性 size maxlength rows cols

2、选择全部文本.select() 选择事件onselect 选择部分文本 设置选择范围

剪贴板事件 3、自动切换焦点（输入达到最大值） 4、表单验证 约束 必填 范围 检查有效性 无效原因：validty对象 不验证

5、select选择框 选中项 移除选项 增加选项 选项列表 select的value等于选项的value或者文本

选项 文本text 值 value 索引index 是否选中selected 6、序列化表单ajax

十六、跨文档消息传递 message事件 postMessage onmessage 只能发到窗口

拖放 事件dragstart drag dragend 进入目标dragenter dragover dragleave 需要允许drop 需要取消默认事件的打开网页

Event.dataTransfor释放数据 传递字符串 setData getData

2、所有元素默认都是可拖动的，但不一定允许放 draggable

十七、错误处理 error.message 范围错误 解析错误 引用错误（浏览器错误） 语法错误 类型错误(不存在的方法) uri格式错误

2、没有异常处理 就会触发erroe事件 image.onerror

3、类型转换错误if中用非布尔 用==号 if()自动转换

数据类型错误 函数传参不能传错类型，应该进行检测 可能调用不存在的方法。访问不存在的属性 类型转换后可能导致类型错误，无法访问属性和方法

通信错误：错误的uri

错误应该记录到服务器

4、调试 log控制态 log页面 抛出 if()throw 断言assert(condition,message) {throw}

十八、XML

创建xml

解析成dom文档

DOMParser

解析错误

XMLSerializer

load

Xpath寻找节点 xpath表达式

XLST转换

十九 E4A html中增加原生xml支持

Xml变成xml对象

Var x = new XML(“”);

可以使用字面量

X=<><>

x.toXMLString 返回字符串

XMLList

命名空间NameSpace

解析和序列化