词法分析

js区分大小写

js变量需要以$或\_,或者字母开头，首字母不能使用数字，后面可以使用数字

类型值和变量

1.数据类型：原始类型（Number、String、Boolean）和对象类型(Object)和特殊原始类型（null、undefined）

2.特殊对象：数组和函数

3.全局对象：Date,RegExp,Error

4.js解析器自己的内存管理机制，自动回收内存

5.js变量无类型，不在函数内的称做全局变量，函数内变量具有函数作用域

6.基本运算符：+-\*/% 和Math对象

7.正负无穷大值：Infinity

8.非数字运算值：NaN

9.js整数范围-2^53~2^53包含边界值

10.浮点数舍入误差0.3-0.2=0.09999999999999998（使用整数进行金融运算）

11.转义符号\

12.正则：

var str = “大唐风场1风机”；

var reg = /风场/d\*风机/g || new RegExp(/风场[0-9]\*风机/g)

reg.test(str) ==> true

13. typeof返回类型值：Number、String、Boolean、Object、Function、undefined

typeof null ==> object

14.对象的比较实际上是引用的比较，两个单独定义的对象，即使值相等两个对象也不相等，只有定义时引用另一个对象时再比较两个对象才相等

15.变量名提升，函数内定义变量时，变量声明提前

表达式和运算符

1. 对象属性未定义时读取返回undefined
2. 0/0 ==>NaN ， 1/0 ==> Infinity
3. str in obj 判断属性或方法是否在对象中，继承属性也返回true
4. obj instanceof Object 判断左侧对象是右侧类的实例
5. 删除对象属性使用delete obj.params
6. Var 定义变量默认值是undefined

对象

1. 判断是否为对象的自有属性obj.hasOwnProprety(‘str’) ,继承属性返回false

类似于str in obj，但是in的继承属性返回true

还有更简单的方法,判断属性方法: obj.str !== undefined

其中in除了判断属性之外，还可以判断出值为undefined的属性

var o = {x:undefined}

o.x !== undefined 🡺false

o.y !== undefined 🡺false

“x” in o 🡺true

“y” in o 🡺false

1. 判断对象是否是另一个对象原型：p.isPrototypeOf(o) ，原型链上也返回true
2. 序列化对象：JSON.stringify() 和 JSON.parse()

数组（156页）

1. 数组尾部添加push,尾部删除pop(),前面添加unshift,前面删除shift()，修改原数组
2. 数组遍历：for/in（无序），forEach,
3. Arr.join(‘,’)将数组分割成字符串，默认逗号分割，是String.split()的反向操作
4. Arr.reverse()数组颠倒顺序
5. Arr.sort(function(a,b){

return a-b //从小到大排序

})

Arr.sort(function(a,b){

return b-a //从大到小排序

})

1. 数组连接：arr.concat() 生成新数组，不改变原数组，不递归遍历数组

[1,2,3].concal(4,[5,[6,7]]) 🡺[1,2,3,4,5,[6,7]]

1. 数组截取：arr.slice(startIndex,endIndex) , 包含startIndex,不包含endIndex

负数表示从后向前数，-1表示最后一个元素

1. Arr.splice(‘插入或删除位置‘,’插入或删除个数’)，第二个参数省略表示知道数组结尾，操作改变原数组
2. Arr.toString()
3. ES5数组遍历：forEach,map(),filter()

every，some,reduce, reduceRight

1. 查询数值的index：arr.indexOf(value) ，反向搜索lastIndexOf

函数

1. This指向：

function fuc(){

//函数调用内的this指向全局对象Window

console.log(this)

}

fuc()

var o = {

say:function(){

//函数属性调用，this指向调用他的对象o

console.log(this)

}

}

o.say()

function Person(){

//构造函数this指向新创建的对象p

console.log(this)

}

var p = new Person()

//call,apply,bind 改变this指向

类和模块（213页）

1. 构造函数首字母大写，构造函数需要通过new 来实例化
2. 定义原型对象：共用原型的方法

1.通过构造函数.prototype重写原型对象，新定义的原型对象不包含construct属性，需要显示定义

构造函数.prototype = {

construct:构造函数

}

2.通过对原型对象进行扩展，不需要显示定义construct

构造函数.ptototype.method = function(){}

正则表达式：

var reg = /^\s\*/ || new RegExp(/^\s\*/)

str.replace(reg,’-’) //正则替换，返回替换后字符串

str.match(reg) //进行分组,返回分组后的数组

reg.test(str) //检测存在,返回boolean

str.search(reg) //返回符合条件字符的起始位置index

1. 特殊字符标点使用反斜线（\）转义

|  |  |
| --- | --- |
| […] | 括号内任意字符 |
| [^…] | 非括号内任意字符 |
| . | 任意字符 |
| \s | 空白符 |
| \d | 数字,等价于[0-9] |
| {n,m} | 至少n次，不超过m次 |
| {n,} | n次或更多次 |
| {n} | n次 |
| ? | 0次或者1次 |
| + | 一次或多次 |
| \* | 0次或多次 |
| | | 选择，或者 |
| () | 分组 |
| ^ | 开头 |
| $ | 结束 |

1. 修饰符i（不区分大小写）、g（全局匹配）、m（多行匹配）

客户端js

1. 同源策略：协议，端口，域名
2. Window对象：

计时器：setInterval，setTimeout

导航： window.location === document.location

window.location.href === location.toString()

location.search.substring(1) === window.location.href.split('?')[1]

location.search = “?page=” + (pagenum+1)

location.replace('http://www.xxx.com/'); 无痕跳转

a jump b replace c, c.history.go(-1) =>a,b.replace为无痕b

history：历史记录，

navigator：浏览器信息

screen：窗口信息

对话框：alert、confirm、prompt

open()与close() ：打开窗口（手动点击才能触发）与关闭open的窗口

iframe：

父级窗口操作，parent替换window 进行操作

parent取iframe的window:document.getElementById(“iframeID”).contentWindow

脚本化文档

1. 获取元素

document.getElementById

document.getElementsByName

document,getElementByTagName

document.getElementByClassName

document.querySelector，document.querySelectorAll

attributes、innerHtml、textContent、

1. 尺度

offsetLeft相对父元素x,offsetTop相对父元素y

clientLeft 、clientTop不包含边框和滚动条，只包含内容和内边距,通常等于border

clientWidth、clientHeight <= scrollWidth、scrollHeight 包含内容和溢出内容

scrollLeft、scrollTop 滚动条位置

1. Document:

document.domain 返回域名地址

document.title 返回title