js八-node

JS数据类型：（编程时的原材料，得用的非常熟）
JS是弱类型的语言，它的数据类型是由值决定的。
var 变量名 = 值

- 基本数据类型：number,string,boolean,null,undefined
- 引用数据类型：对象（object,array,regExp,Date,Math）和函数(function)

number类型的数据

- 定义：小数，整数(正整数，负整数，0)，NaN(not a number)
- 四则运算（+-*/%）失败时
- 将其他数据类型强制转换number类型失败时 Number() parseInt() parseFloat()
  Number()
  true->1 false->0
  null ->0;
  undefined->NaN
  “”->0
  其他字符串，只要有一个字符(不包括小数点)不是数字，则转换结果是NaN
  Number(“10.5”)->10.5 Number(“10.5a”)->10.5

  parseInt() 将字符串中整数部分提取出来，若第一个字符不是数字，则结果是NaN
  parseFloat 将字符串中整数和小数部分都提取出来，若第一个字符不是数字，则结果是NaN

  运算符优先级:
  算术运算符(+-*/%)>比较运算符(>= <= == !=)>逻辑运算符(逻辑与&& 逻辑或||)>赋值运算符(=)

字符串类型

字符串常用方法
charAt() 参数：索引 通过索引找到对应的字符
charCodeAt() 参数:索引 通过索引找到对应字符的ASCII码值
indexOf() 参数:字符 1.查找字符中是否有这个字符，若找不到则返回-1
2.找到，则返回对应字符的索引
lastIndexOf()

substr(n,m) 从索引n截取m个

substring(n,m) 从索引n截取到索引m(包前不包后)
slice(n,m)从索引n截取到索引m (包前不包后) 可以设置负数索引 = 负数索引+length 倒数第几个
toUpperCase() 转换成大写
toLowerCase() 转换成小写
split(“:”) 按照指定的字符，将数组拆分成数组的每一项
split(“”) var str = “abc” str.split(“”) ->[“a”,”b”,”c”]
split() 将字符串完整的放入数组中 str.split()->[“abc”];
split() 参数也可以是正则
replace() 匹配和替换字符
match() 参数：字符串或正则 将匹配的字符串在数组中输出
search() 参数：字符串或正则
trim() 去掉字符串的收尾空格 若想所有浏览器都支持 写个正则匹配收尾空格

es6的字符串方法
includes 查看是否包含某个字符或字符串
startsWith 查看是否以某个字符或字符串开头 默认是索引0开始查找 第二个参数表示设置查找的位置，
endsWith 查看是否以某个字符或字符串结尾 默认查找到最后 第二个参数表示设置的是查找结果的索引（不包含索引这一项）
repeat() 重复某个字符或字符串 参数指重复的次数

boolean

将其他数据类型转换成布尔类型
Boolean() 0,null,NaN,undefined,””->false 其他全部为true
!”a”->!后面的转换成布尔类型，然后再取反
!!”a”->为了转换成布尔类型

null undefined

null 现在没有值但以后会有值 原本应该有值，但没有值
undefined 从来没有存在过

object类型

- 需要从多方面描述事物,对象是多种数据类型的复合载体
- 增删改查 遍历
- 操作时都能以[]或.方式操作
- 若属性名数字只能以[]方式操作
  es6提供的方法

Object.is() 查看对象是否相等(检测是否是指向的同一个引用地址)
Object.is({},{}) false

Object.assign() 合并对象
var obj1 = {name:”lily”};
var obj2 = {age:10};
Object.assign(obj1,obj2);
Object.assign({},obj1,obj2);
Object.assign({},obj1);//拷贝对象
Object.assign(obj1) //还是表示obj1,没有意义
{…obj1,…obj2} 合并对象

数组

es5方法(包含迭代方法)
原有数组改变
push() 往数组的末尾添加内容 返回值: 添加内容后的长度
pop() 删除数组的最后一项 返回值：删除的内容
shift() 删除数组的第一项 返回值：添加内容后的长度
unshift() 往数组的开始位置添加内容 返回值: 添加内容后的长度
splice(n,m,x) 从索引n开始删除m个，用x的内容替换删除的内容 返回值:把删除的内容放在新数组中返回,若没有删除的内容返回值是空数组
reverse() 反向排列
sort(function(a,b){ 排序，不传参表示按ASCII码值进行排序，若传参则按照自定义规则排序
return a-b/b-a
return a.localCompare(b)/b.localCompare(a);
))
原有数组没有改变
join(“&”) 按照指定的字符将数组拼接成字符串
toSting() 将数组转换成字符
concat() 合并数组
indexOf() 查找数组中是否有这一项 若没有则返回-1，若有则返回数组这一项的索引
lastIndexOf()
slice(n,m) 截取数组 从索引n截取到索引m(包前不包后)

迭代方法
forEach() 遍历数组没有返回值
map() 遍历数组并能修改数组的内容有返回值
some() 数组中只要有一个成立，则结果true,只有都不成立才为false
[1,2,3].some(function(item){
return item>1
})
every 都成立才返回true,只要有一个不成立则返回false

[1,2,3].every(function(item){
return item>1
})
reduce() 从左往右求数组的累计的值
reduceRight 从右往往求数组的累计的值

es6方法
find() 返回满足条件这一项,找到了则方法中止运行
[1,2,3].find(function(item){
return item>1
})
findIndex() 返回满足条件这一项的索引,找到了则方法中止运行
[1,2,3].findIndex(function(item){
return item>1
})

Array.from() 将类数组转换成数组
Array.of() 将参数（一个数或一组数）转换成数组
copyWithin(target,start,end) 拷贝数组的部分内容覆盖到指定的位置（会覆盖原来的内容）
target：指拷贝内容放置的起始索引
start :获取内容的起始位置索引
end：获取内容的结束位置索引

遍历数组
for(let value of ary){}
for(let key of ary.keys()){}
for(let ele of ary.entries()){}

函数

函数定义:
1.开辟一个堆内存，假设引用地址是FFF000;
2.将函数体的内容以字符串的形式存在堆内存里
3.将引用地址赋值给函数名，函数名就代表了整个函数
函数执行：
1. 开辟一个私有的作用域
2. 形参赋值
3. 变量提升（对var关键字和function关键字进行声明和定义 es6没有变量提升）
4. 代码从上往下执行

return(返回值):
1.函数遇到return无条件中止运行
2.return可写可不写，根据需求来
3.若没写return 或return后没有值，则函数的返回值是undefined

4.return 后面的不预解释，但是return下面的还是要预解释

私有作用域 - 变量的查找
私有变量:形参，带var关键字
函数内部不带var关键字查找变量的顺序：
```
var num = 10;
function fn(){
console.log(num);
}
fn();
```

- 先看是否是私有的，若是私有的则与外界无关
- 若不是私有的，则往上级作用域查找，若没找到继续往上级查找，直到找到window，若还没找到则报错
  上级作用域跟函数定义有关，跟函数执行无关

this

1. 看函数运行时，前面有没有点，点前面是谁，this就是谁，若没有则this是window
2. 自执行函数中的this是window
3. 事件绑定函数中的this是绑定的元素
4. 定时器中this是window
5. 构造函数中的this是实例
6. 箭头函数中没有this,this需要往上级作用域中查找

原型

- 所有的函数都有个属性叫prototype，指向于原型对象
- 默认的原型上有个属性叫constructor,指向所属的类
- 所有的对象上都有个属性叫**proto**,它指向于所属类的原型
  继承：
  call继承：父类的私有属性
  原型继承:父类的公有的属性 父类私有属性会污染子类的原型
  寄生式组合继承：解决了原型继承的问题

回调函数

定义:将一个函数的定义作为参数传给另一个函数时，这个函数就称为回调函数

- 回调函数的执行次数
- 回调函数的参数

- 回调函数有没有返回值
- 回调函数里的this关键字
  封装map,bind方法封装 敲熟

回流和重绘

DOM映射

js盒子模型

- 13个js盒子模型相关属性
  其中11个只能取读，只有scrollLeft和scrollTop既可以获取可以设置
- 获取任意的css属性(不管是行内，内嵌，外链)
  window.getComputedStyle(ele,null)[attr]
  ele.currentStyle[attr]
  掌握封装getCss,setCss,setGroup,css 敲熟
- 浏览器兼容性处理方案
  - 1.try…catch… 捕获浏览器异常
  - 2.属性判断的方式
    window.getComputedStyle
    “getComputedStyle” in window
  - 3.检测数据类型的方式
    typeof “getComputedStyle” == “function”
    ary instanceOf Array
    Object.prototype.toString.call([])==”[object Array]”
  - 4.constructor

    > 原型对象若被重写，则constructor有可能不准

  - 5.检测浏览器的方式
    navigator.userAgent.indexOf(“MSIE 8.0”)
    /MSIE [6-8].0/.test(navigator.userAgent)

正则

- 掌握常用的元字符
  \d \w \s \b \n .
  \D \W \S \B
  ^ $

(a|z) a或z
[a-z]
[^a-z]
[az]

- 1到多次
- 0到多次
  ？ 0次或1次
  {n} 匹配n次
  {n,}至少匹配n次
  {n,m} 至少匹配n次，最多匹配m次
  g 全文查找
  m 换行查找
  i 忽视大小写

正则 test/exec
字符串 split/replace/match

regExecAll()->把匹配的内容放入数组的返回
1.求出现次数最多的字母和次数（2种）
2.格式化url的queryString部分，把参数放在对象中返回?name=zf&age=9
{name:”zf”,age:9}
3.格式化时间，按照模板返回指定格式的时间

jquery中常用的方法

1.核心
2.ajax
3.效果
4.属性
5.文档处理
6.筛选
7.CSS
8.事件 on off

cookie使用注意事项

cookie是为了辩别用户身份，进行会话跟踪而存储在客户端上的数据

- 可能被客户端篡改，使用前验证合法性
- 不要存储敏感数据，比如用户密码，账户余额
- 使用httpOnly保证安全防止XSS攻击产生
- 设置正确的domain和path，减少数据传输

设置cookie
document.cookie = “key=value”
获取cookie
document.cookie

在express中操作cookie

设置cookie
res.cookie(name,value,options)
获取请求头中的cookie
req.cookies.name

第三方中间件（cookie-parser）来解析cookie

设置cookie
res.cookie(name,value);
获取请求头中的cookie
req.cookies
清除cookie
res.clearCookie(name)

```
//1.普通设置
//res.cookie('name','value');

//2.设置域名
//res.cookie('name','zfpx',{domain:'a.zfpx.cn'});

//3.设置路径
//res.cookie('name','zfpx',{path:'/visit'});

//4.过期时间
//res.cookie('name','zfpx',{expires:new Date(Date.now()+20*100
0)});//毫秒
//res.cookie('name','zfpx',{maxAge:20*1000});//过期时间 毫秒

//httpOnly true还是false无意义 document.cookie取不到
//res.cookie('name','zfpx',{httpOnly:true});
```

第一次 客户端向服务器端发送请求
服务器端接收到请求后，将sessionId保存在客户端cookie中
第二次 客户端再次向服务器发送请求时，会把保存sessionId的cookie发送给服务器端，服务器端通过cookie中的sessionId找到服务器中对应的数据发送给客户端

session

1.什么是session?

- session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而session保存在服务器上
    1. cookie与session区别
       cookie数据存放在客户的浏览器上，session数据放在服务器上。
       cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗 考虑到安全应当使用session
       session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能 考虑到减轻服务器性能方面,应当使用COOKIE
       单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie
       将登陆信息等重要信息存放为session、其他信息如果需要保留，可以放在cookie中
    2. 若sessionId没有存在客户端cookie里，说明是第一次访问或cookie过期，这给设置客户端cookie，并返回{balance:100};
       再次访问时 ，每次余额减20