1801知识点

笔记本： 凡尘：二阶段
创建时间： 2018/3/26 星期一 下午 4:18 **更新时间：** 2018/6/4 星期一 下午 4:56
作者： 437389128@qq.com

001、你对js的理解

a、由网景公司（Netscape）开发

b、这是一个跨平台的脚本语言

c、应用在网页中，用来操作浏览器及网页内容

d、javascript最初的设计是为了检验HTML表单输入的正确性

002、js的三大组成部分

ECMAScript：一种规范

BOM(browser) object Model：浏览器对象模型

DOM(document):文档对象模型

003、js引入的方式

1、

2、写在<script></script>

```
<script>
    alert(1)
</script>
```

3、写在js文件

```
<script src="js/index.js"></script>
```

004、js的注释

单行注释：//

多行注释：/**/

005、js中变量的命名规则

规则：数字、字母、下划线、$符号，其中不能以数字开头

声明：不可以使用关键字和保留字

建议：见名知意

命名方法：驼峰命名法、常规命名(_连接)、匈牙利命名

006、js中的数据类型有哪些

js的数据类型分为：基本数据类型、复杂数据类型

基本数据类型：number string boolean null undefined
复杂数据类型：object

007、null 、 true false转换成数字的值都是多少

false如果转换成一个数字的话是0
true如果转换成一个数字的话就是1
null如果转换成一个数字的话就是0

008、js中的类型转换有哪些

js中的类型转换分为：强制类型转换 隐式类型转换

强制类型转换：Number parseInt parseFloat

隐式类型转换：
  算数运算符：- * / %

  赋值运算符:-= *= /= %=

  关系运算符:== !=

009、谈谈你对NaN的理解

a、Not a Number 不是一个数字

b、NaN仅仅代表不是数字，自身和自身都不相等（NaN != NaN）

c、如何判断NaN:isNaN===>如果是NaN,返回true,否则返回false

d、NaN的数据类型是number

010、说下i++与++i之间的区别

i++: 先使用,在加1
++i; 先加1,在使用

011、 javascript转换成false的值有哪些？

0 '' null undefined false

012、谈谈你对javascript+号的理解

1、任何字符串与加号后面的数据进行相加都会变成链接

2、如果与undefined相加得出的值是NaN

013、null和undefined的区别

null是一个表示"无"的对象，转为数值时为0；undefined是一个表示"无"的原始值，转为数值时为NaN。
undefined：
（1）变量被声明了，但没有赋值时，就等于undefined。
（2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。
（3）对象没有赋值的属性，该属性的值为undefined。
（4）函数没有返回值时，默认返回undefined。
null：
（1） 作为函数的参数，表示该函数的参数不是对象。
（2） 作为对象原型链的终点。

014、if语句的优化

1、把次数多的条件和执行结果放到最前面

2、减少第一次无用的判断，可以用嵌套判断

3、判断语句禁止出现三次嵌套

015、谈谈你对switch的理解

1、switch的括号里面放的是一个变量

2、case相对应的值是关于这个变量的一个值

3、switch里面的这个变量和case里面这个变量不会进行隐式类型的一个转换，而是进行了恒等比较。所以一定要注意这个变量和这个case里面的值是不是一个类型

4、关于switch里面的case会有一个穿透效果，这个效果有的时候会给我们带来好处(详情请看最后一个案例)，有的时候会给我们带来坏处，如果不需要这种穿透效果的时候加break

5、swicth里面如果这个变量没有匹配到case里面这个值，那么就需要返回一个信息。所以在case的末尾一定要加上一个default;这样既给用户的体验比较完美，另一方面对代码的今后维护也有很大的帮助

6、比较的值是固定值

016、if和swicth的应用场景

if :
1、具体的值进行判断
2、区间的判断
3、对运算的结果是boolean类型表达式进行判断 true flash

switch :
1、对具体的值进行判断
2、值的个数是固定的

对于几个固定的值判断，建议使用switch 语句。因为switch 语句会将具体的答案都加载进内存，效率相对高一点
基于代码的可读性：如果条件较少时,if-else容易阅读,而条件较多时switch更容易阅读

017、目前所了解的转义字符

\"
\'
\n 表示换行
\t tab键

018、循环的概念和意义

1、什么是循环？
重复执行某个动作或者某些语句(代码块)

2、循环有什么作用？
a、简单代码，处理重复的动作
b、一般用于遍历数组、json对象、元素集合

019、while需要注意的地方

1、初始化一个变量

2、while()括号里面是判断条件

3、除了在执行相应的代码以外还要在代码块中改变循环体的条件变量===》在{}里面改变

020、do-while循环的使用及while的区别

do-while() 无论条件是否成立至少执行一次，和while规则一样，唯一不同的是do{}while会先执行一次(先执行后判断

021、谈谈你对for循环的理解

```
for(第一个是：循环变量也是初始化变量;第二个是判断循环条件;第三个是改变循环变量){
    ....代码块
}

循环变量： 用于控制循环是否结束的变量(给变量赋初始值,只执行一次)
循环条件: 判断循环是否继续(每次都会执行)
改变循环变量：用于改变循环变量(每次都会执行)
```

022、while和for的区别

for循环是知道了循环次数，while是不知道循环次数

for限定了循环次数

while是条件循环

023、break和continue return的区别

continue:continue只是中止本次循环，接着开始下一次循环 ,只能出现在循环中

break:break用于完全结束一个循环，跳出循环体 不在执行break下面的代码,只能出现在选择或者循环中

return:返回函数的值，不在执行return下面的代码，只能出现函数中

024、什么是一个函数，谈谈你对函数的理解

官方解释：由事件驱动的或者当它被调用时执行的可重复使用的代码块

白话：函数就是一个方法、一个功能

025、让函数执行有哪几种方式

1、js是基于事件驱动的语言、因此可以通过js的事件来调用函数让函数执行

2、直接写函数名加()

3、立即执行函数。在字面量函数后面({}后面加())

026、定义函数的几种方式

1、关键字函数：function fnName(){};

2、字面量函数：var fn = function(){};

3、构造函数：var fn = new Function()

027、函数的作用

1、减少代码的编写(代码重复利用)

2、隐藏处理细节，便于今后的修改和维护

3、控制执行时机

028、谈谈你对参数的理解

1、参数分为：形参和实参

2、有了参数以后可以使函数变的更加灵活

3、形参和实参要一一对应

4、如果对应的形参没有传值，那么值是undefined

029、什么是作用域链？

1、简单说就是作用域集合 当前作用域 -> 父级作用域 -> ... -> 全局作用域 形成的作用域链条

030、console.log与console.dir的区别

console.log()可以取代alert()或document.write()，在网页脚本中使用console.log()时，会在浏览器控制台打印出信息。

console.dir()可以显示一个对象所有的属性和方法。

031、谈谈你对arguments的了解

1、函数内部自带的一个对象

2、存储的是所有的实参

3、可以使用[]及下标访问arguments中的内容 arguments[0] 访问第一个实参

4、可以使用 arguments.length 确定传入实参的个数

5、最常用的用途： 判断传入参数的个数(根据参数个数做不同的事情)

032、请说一下js的编译和执行

1、js的预编译：
   a、把var 和 function 定义的变量提升到script的最上方
   b、赋值语句不会被提升，哪怕等号后面是一个function
2、js执行：代码从上往下执行

033、简单的阐述一下js的变量声明提升

变量声明和函数声明从他们代码中出现的位置被移动到执行环境的顶部，这个过程就叫做提升 只有声明操作会被提升，赋值和逻辑操作会被留在原地等待执行

Js编译器会把变量声明看成两个部分分别是声明操作(var a)和赋值操作(a=2)

034、简述你对js中堆栈的了解

js变量存储有栈存储和堆存储，基本数据类型的变量存储在栈中，引用数据类型的变量存储在堆中 引用类型数据的地址也存在栈中

当访问基础类型变量时，直接从栈中取值。当访问引用类型变量时，先从栈中读取地址，在根据地址到堆中取出数据

当

035、对象类型的好处在哪？

使得信息的传递更方便快捷

036、构造函数是干什么用的？

构造函数是初始化已创建好的对象中成员变量的

背会！！！！递归与循环的区别

递归算法：

优点：代码简洁、清晰，并且容易验证正确性。

缺点：
1、它的运行需要较多次数的函数调用，如果调用层数比较深，每次都要创建新的变量，需要增加额外的堆栈处理，会对执行效率有一定影响，占用过多的内存资源。
2、递归算法解题的运行效率较低。在递归调用的过程中系统为每一层的返回点、局部变量等开辟了栈来储存。递归次数过多容易造成栈溢出等

注意：递归就是在过程或函数里调用自身；使用递归策略时要注意的几个条件
1、必须有一个明确的递归结束条件，称为递归出口。
2、递归需要有边界条件、递归前进段和递归返回段。
3、当边界条件不满足时，递归前进。当边界条件满足时，递归返回。

循环算法：

优点：速度快，结构简单。

缺点：并不能解决所有的问题。有的问题适合使用递归而不是循环。如果使用循环并不困难的话，最好使用循环

037、数组的概念及作用

概念：一组(一般情况下相同类型)的数据（不一定都是数字）

作用：就是一个容器，将多个数据保存起来

038、创建数组的几种方式

三种方式：
1、var arr = new Array();
2、var arr = Array
3、var arr = [];

039、如果判断一个对象是不是数组

数组.instanceof Array

040、数组常用的一些方法

1、push: 在数组最后添加一个或者多个元素，返回添加后数组的长度

2、pop: 从数组最后取出一个元素，返回的是数组的最后一个元素（取出的元素）

3、unshift: 和push相反，从数组的第一位的前面开始添加

4、shift: 和pop相反从数组的第一位开始取，返回取出的值

5、sort 排序方法

6、reverse 反转

上面的6种方式都在原数组进行操作，会改变原本的数组

041、如何将数组转换为字符串？如何将字符串转换为数组

```
var str = 数组.join('')
var arr = 字符串.split('')

除此之外还有toString()
```

042、slice与splice的区别

```
slice只能截取一段字符串


splice:可以替换  插入   删除
```

043、简单谈一谈关于值传递与引用传递

```
 值传递：传递的是基本数据类型的数据（数据不会发生改变）

 引用传递：传递的是对象（数组、对象）对象存储在堆空间中，自身可以发生改变
```

044、关于数组的排序请用至少2种方式进行排序

```

```

045、什么是json?

```
JSON(JavaScript Object Notation, JS 对象标记) 是一种轻量级的数据交换格式,用来来存储和表示数据

语法：
    [
    {键:值},
    {键:值},
    {键:值},
    ]
```

046、什么是ES5?js中的'use strict'是什么？目的是什么？

```
ECMA Script5：ECMA的第五次改版  时间：2009年

'use strict'js的严格模式

目的：
增加更多报错的场合，消除代码运□的□些不安全之处，保证代码 运行的安全。
提高编译器效率，增加运□速度。
为未来新版本的JavaScript做好铺垫
```

047、ES5中新增的数组的方法有哪些？

```
indexOf()：返回第一次出现的下标

lastIndexOf()：返回最后一次出现的下标

forEach()：循环

map():映射

filter():过滤
```

048、ES5中字符串新增的方法有哪些？

charAt() 获取指定位置上的字符

indexOf() 获取字符第一次出现的位置

substring() 截取字符串内容

slice() 截取

split() 将字符串分割成数组

replace() 替换

substr 返回一个从指定位置开始的指定长度的子字符串

049、如何将一个字符转换成ascii码？如何将一个数字转换成对应的字符

charCodeAt()：将字符转换成ascii码

String.fromCharCode()：将数字转换成对应的字符

050、请列举出Math常见的API

1、Math.random()=====随机数

2、Math.pow()====次方

3、Math.round()====四舍五入

4、Math.ceil() ===向上取整

5、Math.floor()===向下取整

6、Math.PI ====π

7、Math.max()===返回最大值

8、Math.min()===返回最小值

9、Math.sqrt() ===开方

051、在js中如何用方法将10进制的字符转换成16进制和8进制

```
数字.toString(16)
数字.toString(8)
```

052、如何创建一个对象、增加、删除

第一种创建方式：var obj = new Object()

第二种创建方式：var obj = {};

增加属性：obj.属性名　obj.方法名

删除属性：delete obj.属性名　 delete obj.方法名

053、如何创建时间对象

```
new Date()
```

054、如何创建未来或者过去的时间对象

```
var d = new Date('2017-10-20 22:22:22')

var d = new Date('2017/10/20 11:11:11')
```

055、如何获取时间戳

```
getTime();
```

056、常用的时间API

```
    set/getFullYear()====获取年份

    set/getMonth()=====获取或者设置月份  月份是从0开始的

    set/getDate() ====获取或者设置日期

    set/getHours()====获取或设置时

    set/getMinutes()===获取或设置分

    se/getSeconds() ===获取或设置秒

    getDay();====如果是星期日的话返回是0;
```

057、如何将日期格式转化成为字符串

```
function dateString(date,sign){
    if(sign==undefined){
        sign='/'
    }
return d.getFullYear()+sign+d.getMonth()+sign+d.getDate()+''+'周'+d.getDay()+d.getHours()+':'+d.getMinutes()+':'+d.getSeconds()
}
```

057、将指定格式字符串转化为日期两种方式

```
getTime()方法是把一个date对象转换为毫秒数

d.parse()方法是把一个日期格式的字符串转换为毫秒
```

058、js中异步的代码有哪些?

```
定时器 延时器

事件

ajax
```

059、url的组成部分有哪些

060、什么是浏览器缓存？

浏览器缓存（Browser Caching）是为了加速浏览，浏览器在用户磁盘上对请求过的文档进行存储，当访问者再次请求这个页面时，浏览器就可以从本地磁盘显示文档，这样就可以加速页面的阅览

061、如何打印当前浏览器的版本等信息

navigator.userAgent

返回包含浏览器版本等信息的字符串 ，常用于判断浏览器版本及使用设备（PC或者移动端

062、window.onload与window.onscroll的区别

window.onload:当页面加载完成的时候

window.onscroll:当页面滚动的时候

063、请简述下关于DOM的增、删、查

```
增---------createElement
var oDiv = document.createElement('div');
document.body.appendChild(oDiv);//只有追加后，页面上才会出现

2、删---------removeChild
语法:fatherObj.removeChild(childrenObj)

参数解释：
a、fatherObj:要删除子元素的元素对象
b、childrenObj:要被删除的子元素对象

3、查---------六种
document.getElementById('');常用

document.getElementsByClassName('');(ie9+)常用

document.getElementsByTagName('');常用

document.getElementsByName('');不常用

document.querySelector(选择器)(IE8+)(IE8+)
    根据选择器返回匹配到的第一个元素

document.querySelectorAll(选择器);(IE8+)常用
    根据选择器返回匹配到的所有的元素
```

064、请简述下关于Dom节点的分类

childNodes：获取所有节点 包括文本节点

节点分为3种类型：

1、元素节点

2、文本节点 xxx

3、属性节点

可以通过nodeType这个属性查看节点的类型
nodeType == 1 元素

nodeType == 2 属性

nodeType == 3 文本节点

065、获取非行间样式

```
function getStyle(obj,attr){
    if(obj.currentStyle){
        return obj.currentStyle[attr];
    }else{
        return getComputedStyle(obj,false)[attr];
    }
}
```

066、如何获取、设置、删除自定义属性

```
元素.getAttribute(属性名);

元素.setAttribute(属性名,属性值);

元素.removeAttribute(属性名);
```

067、关于IE的calss与className

FF:使用setAttribute("class", vName)语句动态设置Element的class属性在firefox中是行的通的

IE:在IE中却不行。因为使用IE内核的浏览器不认识"class"，要改用"className"

068、innerHTML、innerText和outerHTML、outerText的区别(了解一下 背会就行不需要抄写)

- innerHTML 设置或获取位于对象起始和结束标签内的 HTML

- outerHTML 设置或获取对象及其内容的 HTML 形式

- innerText 设置或获取位于对象起始和结束标签内的文本

- outerText 设置(包括标签)或获取(不包括标签)对象的文本

innerText和outerText在获取时是相同效果，但在设置时，innerText仅设置标签内的文本，而outerText设置包括标签在内的文本。

不同之处:
  简单的说innerHTML和outerHTML、innerText与outerText的不同之处在于：
    1）、innerHTML与outerHTML在设置对象的内容时包含的HTML会被解析，而innerText与outerText则不会。
    2)、在设置时，innerHTML与innerText仅设置标签内的文本，而outerHTML与outerText设置包括标签在内的文本。
  3）、outhtml innerText(非W3C)

069、如何进行DOM节点筛选

1、可以通过nodeType
1 代表元素节点 2 代表属性节点 3 代表文本节点

```
function getNode(ele){
    var aNode = ele.childNodes;
```

```
    var newArr = [];
    for(var i=0;i<aNode.length;i++){
        if(aNode[i].nodeType == 1){
            newArr.push(aNode[i])
        }
    }
    return newArr;
}
```

070、 childNodes与children、attributes 的区别

如果想获取到子元素的element节点，最好使用children方法，childNodes方法以及firstChild方法在现代浏览器中使用，都会把元素标签中的空白节点检测出来，一般我们使用这两个方法都是为了获取到元素的元素节点，空白节点会给我们造成很多不必要的bug,而children方法则是只检测element元素节点

attributes：获取当前元素节点的所有属性节点集合

071、如何获取父级节点、上一个子级节点、下一个子级节点

nextElementSibling 后一个兄弟元素 （如果没有是null）

previousElementSibling 前一个兄弟元素 （如果没有就是null）

parentNode 获取当前节点的父节点

072、元素节点的创建、添加、删除、替换、克隆

1、创建：document.createElement('元素名');

2、添加：appendChild()====>插入到最后　　insertBefore()====>插入到最前面
用法：

```
function append(obj,newEle){
    var achildren = obj.children;
    if(achildren.length>=1){
      return  obj.insertBefore(newEle,achildren[0])
     } else{
      return   obj.appendChild(newEle)
     }
}
```

3、删除：box.removeChild(node) 从元素中移除某个子元素
用法：　先找到父级节点然后在去删除子级节点

4、替换：box.replaceChild(子元素，父元素)

用法：
父元素.replaceChild(被替换成的元素,替换元素)

5、克隆：clone()
元素.clone();
如果里面传true的话会吧整个标签的所有节点都克隆，如果没有加true,只会克隆当前元素

073、浅谈关于文档碎片的理解

1、js操作dom时发生了什么？

每次对dom的操作都会触发"重排"，这严重影响到能耗，一般通常采取的做法是尽可能的减少dom操作来减少"重排"

2、什么是文档碎片？

document.createDocumentFragment()
一个容器，用于暂时存放创建的dom元素

（其实这个跟咱们上课说那个先让到一个元素中，然后最后appendchild一样）

3、文档碎片有什么用？

将需要添加的大量元素 先添加到文档碎片 中，再将文档碎片添加到需要插入的位置，大大减少dom操作，提高性能

074什么是回流和重绘

当渲染树中的一部分或者全部因为元素的尺寸、布局、隐藏等改变而需要重新构建的时候，这时候就会发生回流。

每个页面都至少发生一次回流，也就是页面第一次加载的时候。

在回流的时候，浏览器会使渲染树中受到影响的元素部分失效，并重新绘制这个部分的渲染树，完成回流以后，浏览器会重新绘制受到影响的部分元素到屏幕中，这个过程就是重绘

什么时候会发生回流？

1、添加或者删除可见的DOM元素的时候

2、元素的位置发生改变

3、元素尺寸改变

4、内容改变

5、页面第一次渲染的时候

075、offsetWidth、offsetHeight 、offsetLeft、offsetTop、offsetParent的区别

offsetWidth、offsetHeight:获取当前元素大小，是按照盒模型获取的，而我们封装的getStyle(obj,attr)是获取的元素本身的大小，没有按照盒模型获取 ，你可以理解成怪异盒模型和标准盒模型

offsetLeft、offsetTop、offsetParent

offsetLefr和offsetTop是获取当前元素距离第一级拥有定位元素属性的父元素的距离

offsetParent:是元素的第一级拥有定位属性的父元素

076、关于offsetX、offsetY、clientX、clientY、pageX、pageY、screenX、screenY的区别

offsetX、offsetY:
鼠标相对于事件源元素（srcElement）的X,Y坐标

clientX、clientY:
鼠标相对于浏览器窗口可视区域的X，Y坐标（窗口坐标），可视区域不包括工具栏和滚动条。

pageX、pagey:
类似于event.clientX、event.clientY，但它们使用的是文档坐标而非窗口坐标。这2个属性不是标准属性，但得到了广泛支持。IE事件中没有这2个属性

screenX、screenY:
鼠标相对于用户显示器屏幕左上角的X,Y坐标。

077、怎么获取元素距离文档边界的值

```
function offset(ele){
    var obj = {};
    obj.l = ele.offsetLeft;
    obj.t = ele.offsetTop;

    while(ele.offsetParent){
      ele = ele.offsetParent;
```

```
        obj.l+=ele.offsetLeft;
        obj.t+=ele.offsetTop;
      }

     return obj;
}
```

078、如何在当前视窗的可见范围看见当前元素

```
scrollIntoView()
```

079、关于onkeydown和onkeypress的区别以及如何获取按下键盘的键盘码？

```
onkeydown:
        1、所有的英文字符都是大写
        2、功能键也会被识别
onkeypress
        1、所有英文字符大小写都支持
        2、所有功能键都不会被识别
        3、组合键ctrl+回车的code值是10


e.keyCode || e.which
```

80、什么是事件流？ 什么是事件冒泡？ 什么是事件捕获？

```
什么是事件流？

    当一个HTML元素产生一个事件时,该事件会在元素节点与根节点之间的路径传播，路径所经过的节点都会收到该事件，这个传播的过程叫做DOM事件流

元素触发事件时，事件的传播过程称为事件流，过程分为捕获和冒泡两种

冒泡事件：微软提出的  事件由子元素传递到父元素的过程，叫做冒泡

捕获事件：网景提出的  事件由父元素到子元素传递的过程，叫做事件捕获
```

81、什么是浏览器默认行为？如何组织浏览器默认行为？

```
例如：右键菜单  点击超链接跳转  图片拖拽

组织默认行为

    return false

    preventDefault是一个方法：e.preventDefault();

    returnValue的值是一个常量：e.returnValue = false;
```

82、如何组织右键菜单？

```
document.oncontextmenu = function(e){

    e.preventDefault()

}
```

83、如何组织事件冒泡

```
利用事件对象属性：stopPropagation 和 cancelBubble
```

stopPropagetion是一个方法：e.stopPropagetion();

cancelBubble的值是一个常量：e.cancelBubble = true;

84、什么是事件监听、事件绑定、事件委托？

事件监听：为同一个对象的同一个事件绑定多个事件处理程序

事件绑定：对DOM元素绑定事件处理函数 一般分为三种 1、在DOM元素中直接绑定 2、在js中绑定 3、绑定事件监听函数

事件委托：利用冒泡的原理，把事件加到父级上，触发执行效果

事件委托的好处：
1、实现对未来元素事件的绑定
未来元素：绑定事件时，页面上还不存在的元素

2、减少事件绑定，提高性能

85、事件监听的方法及如何解绑事件

事件监听方法：

addEventListener() attachEvent()

解绑事件

removeEventListener() detachEvent()

86、谈谈你对事件监听的理解（最好背会，因为太多就适当抄写3遍意思意思吧）

1、事件分为DOM 0级事件和Dom 2级事件，DOM2级事件也叫做事件监听。DOM 0级事件的缺点是如果事件相同 后者的事件会覆盖前者的事件，DOM2级事件可以解决这个问题

2、DOM2级事件的方法是

addEventListener()
参数1：事件类型 不需要加on
参数2：回调函数
参数3：布尔值 true代表捕获 false代表冒泡

解绑事件方法：removeEventListener()

但是IE不支持此方法
IE浏览器下用：attachEvent()
参数1：事件类型 需要加on
参数2：回调函数

解绑事件方法：detachEvent()

3、事件流、事件冒泡、事件捕获

当一个HTML元素产生一个事件时,该事件会在元素节点与根节点之间的路径传播，路径所经过的节点都会收到该事件，这个传播的过程叫做DOM事件流

元素触发事件时，事件的传播过程称为事件流，过程分为捕获和冒泡两种

冒泡事件：微软提出的 事件由子元素传递到父元素的过程，叫做冒泡

捕获事件：网景提出的 事件由父元素到子元素传递的过程，叫做事件捕获

4、IE与火狐的事件机制有什么区别?
事件处理机制：IE是事件冒泡、火狐是 事件捕获;

5、事件代理/事件委托
　利用冒泡机制，将子元素的事件委托给父元素去监听(给父元素添加事件)，当子元素触发事件时，事件冒泡到父级如果希望指定的子元素才能触发事件，可以通过事件对象(event)获得事件源(target)，然后通过　　条件判断是不是期望的子元素，如果是的话，执行事件，否则不执行

6、事件委托的好处
　1、实现对未来元素事件的绑定

　2、减少事件绑定，提高性能

7、如何找到事件源
　var target = e.target||e.srcElement

　　tagName能找到事件源的元素名

87、什么是cookie?

会话跟踪技术

特点：

1、大小限制(不能超过4K)

2、每个域下cookie不能超过50个

3、有效期（和设定时间有关），过了有效cookie会自动删除

4、cookie读取（只能访问同一个域名下的cookie）

5、cookie只能是字符串（文本文件）

88、封装cookie 获取cookie 设置cookie

```
//设置cookie
function setCookie(_name,_val,_expires){
    var d = new Date();
    d.setDate(d.getDate()+_expires);
    document.cookie = _name+'='+_val+';path=/;expires='+d.toGMTString();
}

//删除cookie
function removeCookie(_name,_val){
    setCookie(_name,_val,-1);
}


//获取指定cookie

function getCookie(_name){
    var str = document.cookie;
    var arr = str.split('; ');
    var len = arr.length;
    for(var i=0;i<len;i++){
        var newarr = arr[i].split('=');
        if(newarr[0]==_name){
            return newarr[1]
        }
    }
}
```

89、什么是正则

0、一种高效处理字符串的规则，又称规则表达式

1、正则表达式(regular expression)描述了一种字符串匹配的模式

2、正则表达式是由普通字符（例如字符 a 到 z）以及特殊字符（称为"元字符"）组成

90、为什么要使用正则？正则可以做什么？

1、普通表单验证方法太复杂，效率低下

2、正则写法简单，灵活，效率高，使用非常方便

3、正则表达式可以做字符串的：查找 匹配 替换 提取

91、创建正则表达式？区别

字面量形式：//

构造函数：var reg = new RegExp(表达式,参数);

区别：
1、字面量方式中出现的一切都是元字符,所以不能进行变量值的拼接，而实例创建的方式是可以的。

2、字面量中直接写\d就可以，而在实例中需要把它转译\\d

92、ES6中let、const有哪些特点？

let:
1、变量不允许被重复定义
2、不会进行变量声明提升
3、保留块级作用域中i的值

const:
1、常量值不允许被改变
2、不会进行变量声明提升

93、简单阐述ES6的箭头函数与普通的函数区别？

1、书写上用=>代替了function

2、普通函数的this指向window 而ES6箭头函数里面的this指向定义时的那个对象 而不是运行时的那个对象

94、ES6的Symbol的作用是什么？

ES6引入了一种新的原始数据类型Symbol，表示独一无二的值

95、ES6中字符串和数组新增了那些方法

字符串
1、字符串模板
2、includes
3、startswith
4、endsWith 等

数组
1、Array.of

2、Array.from 等

96、谈谈你对ES6中set集合和Map集合的理解？

set:
set是ES6提供的一种新的数据结构，类似于数组，但是成员的值是唯一的没有重复的，接受的参数是一个数组
方法有：

add():添加
delete():删除
size:长度
has():查找
clear：清除所有

map：
map类似于对象，也是键值对的集合， 但是“键”的范围不限于字符串，各种类型的值（包括对象）都可以当作键
方法有：
set():设置
get():获取
delete():删除
has():查找
clear():清除所有

97、https 与 http的端口号

http:80

https:443

98、构造函数与普通函数 箭头函数的三种区别

1、构造函数的this指向实例化后的那个对象

2、普通函数的this指向window

3、箭头函数的this指向创建时的那个对象，而不是引用时的那个对象

99、谈谈你对面向对象的理解

1、使用对象的时候只关注对象提供的一些功能，不关注内部的一些细节

2、面向对象是一种通用的思想，并非只有编程中能使用。任何事情都可以使用

3、面向对象的三大基本特征
a、抽象：抓住核心问题

b、封装：不考虑内部实现，只考虑功能使用

c、继承：从已有的对象上继承出新的对象

4、 面向对象思维主张的是：团队---配合---分工---协作，将大问题拆分成诺干个小问题，并试图用分工协作来完成

100、面向对象的好处

1、开发时间短，效率高，可靠性高，所开发的程序更强壮。由于面向对象编程的可重用性，可以在应用程序中大量采用成熟的类库，从而缩短了开发时间。

2、应用程序更易于维护、更新和升级。继承和封装使得应用程序的修改带来的影响更加局部化。

101、prototype的作用

1、节约内存

2、扩展属性和方法

3、可以实现类的继承

102、在执行new的过程中js执行了哪些操作？

1、在内存中开辟了一块空间

2、把this指向了当前对象

103、php服务端如何接受客户端传递的数据

$_GET[""]

$_POST[""]

$REQUEST[""]

104、php如何操作数据库

1、设置数据源

2、连接数据源 new mysqli()

3、设置编码格式 $conn->query("set names utf8");

4、编写sql语句

5、执行sql语句 mysqli_query($conn,$sql). delete update insert 返回的都是受影响的行数

6、获取结果集中的数组 mysqli_fetch_assoc()

7、判断结果集中的数据长度 mysqli_num_rows()

105、ajax的优点

a、提高运行效率

b、提高用户体验，让多件事情同时发生

c、在不刷新页面的情况下可以对局部数据进行加载和刷新

106、ajax的工作原理

运行原理：相当于创建了一个请求代理，通过代理去完成与服务器的交互，交互的过程中客户不需要等待，还可以进行其它的工作，交互完成以后，代理再将交互的结果返回给客户页面。

107、ajax请求的流程

1、创建通信对象

a、IE7及其以上版本中支持原生的 XHR 对象，因此可以直接使用
var xhr = new XMLHttpRequest()

b、IE6及其之前版本中,XHR对象是通过MSXML库中的一个ActiveX对象实现的
var xhr = new ActiveXObject("Microsoft,XMLHTTP");

2、链接和发送

a、open() 函数参数有三个：请求方式，请求地址，是否异步请求（同步请求的情况特别少）
xhr.open('get','http://www.baidu.com',true)

b、GET 请求方式是通过URL参数将数据提交到服务器的，POST则是通过将数据作为 send的参数传递

c、xhr.send() 发送请求

3、监听服务器是否返回数据

a、使用onreadystatechange事件监听服务器返回状态

xhr.onreadystatechange = function(){

```
}
```

108、http的状态有哪些(常用的)？(xml.status)

100——客户必须继续发出请求

101——客户要求服务器根据请求转换HTTP协议版本

200——交易成功

304——客户端已经执行了GET，但文件未变化

404——没有发现文件、查询或URl

500——服务器产生内部错误

505——服务器不支持或拒绝支请求头中指定的HTTP版本

109、ajax的状态值有哪些？（xml.readyState）

0-未初始化，尚未调用open()方法

1-启动，调用open()方法，未调用send()的方法

2-发送，已经调用send()方法，未接受到响应

3-接受，已经接受到部分响应数据

4、完成，已经接受到全部响应数据;（我们都是使用xhr.readyState == 4 判断ajax请求是否结束）

110、在IE浏览器下如何解决传递参数乱码问题

encodeURI()

111、什么是同源策略？

同源指的是域名、协议、端口号相同

同源策略规定了js代码的访问权限，只能访问和自己同源的页面。

同源策略是一种约定，它是浏览器最核心也最基本的安全功能

112？什么是跨域？如何解决跨域？ jsonp的原理？

1、由于浏览器的同源策略，即属于不同域的页面之间不能相互访问各自的页面内容。

2、CORS和jsonp

3、利用浏览器的"漏洞" src不受同源策略的影响，可以请求任何链接 。动态创建script标签，将事先写好的函数名传给服务器，供服务器使用

113、什么是jsonp?

jsonp是一种非正式传输协议，用于解决跨域问题

114、jsonp跨域的流程

1、创建一个全局函数

2、创建一个script标签

3、给script添加src

4、给src添加回调函数test(callback=test) callback是传给后端的一个参数

5、将script放到页面上

6、script请求完成，将自己从页面上删除

115、简术你对promise的理解

1、什么是promise?
异步操作的同步代码

2、promise的基本使用

通过new promise创建一个promise对象，里面有一个参数，参数是一个回调函数，回调函数中有2个参数，resolve，reject resolve()当异步执行成功的时候调用的方法，reject()当异步失败的时候调用的方法。

除此之外promise有一个then方法，当成功的时候执行第一个回调函数，当失败的时候执行第二个回调函数。第二个回调函数也可以通过promise对象.catch调用

116、如何实现多个异步同步执行

```
var p1 = new Promise(function(resolve,reject){
    setTimeout(function(){
        console.log('1');
        resolve()
    },3000)
})

function p2(){
    return new Promise(function(resolve,reject){
        setTimeout(function(){
            console.log("2");
            resolve();
        },2000)
    })
}


function p3(){
    return new Promise(function(resolve,reject){
        setTimeout(function(){
            console.log("3");
            resolve();
        },1000)
    })
}

function p4(){
    return new Promise(function(resolve,reject){
        setTimeout(function(){
            console.log("4");
            resolve();
        },500)
    })
}

p1.then(function(){
    return p2()
})
.then(function(){
    return p3();
})
```

```
.then(function(){
    return p4();
})
```

117、请说出call、apply、bind的区别

1118、什么是闭包？用途？注意的地方？

1、闭包就是可以读取其他函数内部变量的函数

2、可以读取函数内部的局部变量 2、让这些变量始终保持在内存当中

3、由于闭包会使得函数中的变量都被保存在内存当中，内存会消耗很大，所以不能够滥用闭包，否则会造成网页性能的问题

119、请简述prototype、__proto__ constructor三者的关系

1、prototype：
每一个函数都有一个prototype这个属性，而这个属性指向一个对象，这个对象我们叫做原型对象

作用：
a、节约内存

b、扩展属性和方法

c、可以实现类之间的继承

2、__proto__

1、每一个对象都有一个__proto__属性

2、__proto__指向创建自己的那个构造函数的原型对象

3、对象可以直接访问__proto__里面的属性和方法

3、constructor：
指向创建自己的那个构造函数

总结：
当我们创建一个构造函数的时候这个构造函数自带了一个prototype属性，而这个属性指向一个对象，也就是原型对象。

这个原型对象里面有一个constructor构造器，它的作用是指向创建自己的构造函数。除此之外prototype还可以存放公共的属性和方法。

当我们实例化一个对象的时候，这个对象自带了一个__proto__属性，这个__proto__指向创建自己的构造函数的原型对象。可以使用这个原型对象里面的属性和方法

120、请写出方法继承的方式

1、call 、apply:不建议使用浪费内存

2、原型对象继承

3、原型拷贝继承

4、原型链继承

5、混合继承

6、继承继承

7、ES6 class super()

121、设计模式

1、单例模式：
保证程序中，使用该模式的类只有一个实例

代码....

2、代理模式：

代码....

3、观察者模式：
某个人--->观察某件事件---》事情发生变化---》通知这个人---》去做某件事情

代码....

122、window.onload、window.onresize、window.onscroll、$(document).ready(function(){})四者的区别

1、window.onload:当文档加载完毕以后　包括html,js,img,css

2、window.onresize：当窗口发送改变的时候,高频率出发事件

3、window.onscroll:当滚动条滚动的时候，高频率出发事件

4、$(document).ready(function(){})

123、jquery中attr() 与prop ()的区别

attr:主要是对元素的属性进行

增: attr("data-id","01");

删:removeAttr("data-id");

改:attr("data-id","02");

查:attr("data-id");

prop：主要是设置元素的 selected/checked

124、jquery中动画的形式有哪几种？

1、基本动画
show()

hide()

toggle()

2、滑动动画
slideUp()

slideDown()

slideToggle()

3、淡入淡出动画
fadeIn()

fadeOut()

fadeTo()

fadeToggle()

4、自定义动画

animate()

125、如何停止动画和延迟动画

.stop()：停止动画

.delay():延迟动画

126、jquery中节点的操作有哪些？

children():获取所有子节点

find():查找后代元素

next():下一个元素

prev():上一个元素

parent():获取父级元素

siblings():获取同级元素

127、jquery中ajax的书写方式及参数

```
$.ajax({
    type:请求方式,
    url:"接口地址",
    data:"传递的参数",
    dataType:"数据类型",
    success:成功的回调,
    error:失败回调

})
```

128、jquery中 width()、innerWidth、outerWidth的区别

width():只会获取content内容区的宽度

innerWidth():会获取content+padding的宽度

outerWidth():会获取content+padding+border的宽度

129、jquery中offset()、position()、scrollTop()

offset():获取当前元素距离页面之间的偏移量

position():获取当前元素距离以定位的父元素的偏移量

scrollTop():获取滚动条滚动的距离

130、$(document).height() 与 $(window).height()的区别

$(documemt).height()：获取整个页面的高度 类似于原生js里面的document.body.clientHeight

$(window).height():获取可视区的高度 类似于原生js里面的document.documentElement.clientHeight

131、jquery中事件绑定、委托中 bind() live() delegate()、on()之间的区别

bind:为每个元素绑定事件处理函数

解绑事件：unbind()

缺点：
　无法对未来元素实现事件绑定

live:为所有匹配的元素添加事件处理函数，未来元素也可以绑定事件处理函数

解绑事件：die()

缺点：
　1、1.7版本后不再支持该方法

　2、阻止事件冒泡不管用

delegate:为指定的元素(子元素)添加一个活多个事件处理函数

解绑事件：undelegate()

在某些浏览器下是有兼容性问题

on: 为匹配的元素绑定一个活多个事件处理函数

解绑事件：off()

缺点：
　on不能取代live

132、如何对一个jquery对象扩展方法？

$.extend():扩展jquery对象本身

例：
```
$.extend({
    min:function(a,b){
        return a>b?b:a;
    },
    max:function(a,b){
        return a>b?a:b;
    }
})

$.min();
```

133、如何对一个jquery元素扩展方法？

$.fn.extend():扩展jquery元素方法

例：
```
$.fn.extend({
    zyh:function(info){
        return this.each(function(){this.innerHTML = info})
    }
})

$("p").zyh("你好");
```

134、jquery中 trigger()的作用及用法

trigger:触发元素身上的某类事件

例：

```
$("p").on("zyh",function(){
    alert(1)
})

$("p").trigger("zyh");
```

简单来说trigger就是自动执行某一个事件不需要手动触发

135、jquery中$.proxy()的作用及用法

$.proxy():返回一个新的函数，并且这个函数始终保持了特定的作用域

参数1：要改变作用域的函数

参数2：一个Object，参数1的作用域会被设置到这个Object上面

作用：可以解决this的指向问题，以及回调函数嵌套的问题

136、$(this) 和 this 关键字在 jQuery 中有何不同

$(this) 返回一个 jQuery 对象，你可以对它调用多个 jQuery 方法

而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素。你不能对它调用 jQuery 方法

137、jQuery 中的方法链是什么？使用方法链有什么好处

方法链是对一个方法返回的结果调用另一个方法，这使得代码简洁明了，同时由于只对 DOM 进行了一轮查找，性能方面更加出色。

138、哪种方式更高效：document.getElementbyId("myId") 还是 $("#myId")？

第一种，因为它直接调用了 JavaScript 引擎。

139、JQ中find()、has()和filter()区别？

filter()方法，条件作用于自身

has()方法条件是作用于它的后代元素中

find():当前选中元素的上下文中找到符合条件的后代，返回的是子元素

140、模块化开发的优点

1、解决文件之间的依赖关系

2、避免命名冲突 、解决全局变量级全局函数泛用的现象

3、解决代码的复杂性

4、按需加载

141、模块化分为哪几种？

1、服务端 (commonJS) 例如 Node.js

2、客户端(AMD CMD) 例如 require sea.js

3、ES6：module(export import);

142、什么是require.js作用是什么？

RequireJS是一个JavaScript文件或者**模块的加载器**。它可以提高JavaScript文件的加载速度，避免不必要的堵塞。

requireJS的作用：
1、实现js的异步加载
2、管理模块之间的依赖关系，便于代码的编写和维护

143、什么是AMD？

AMD是一种**定义和加载模块的规则**，使模块和它的依赖可以被异步的加载，但又按照正确的顺序

144、AMD与CMD的区别

AMD 推崇**依赖前置**（事先加载好需要用到的模块）
CMD 推崇**就近**（用到时再加载）

145、什么是css预处理器？优点是什么？

css预处理器用一种专门的编程语言，进行web页面样式设计，然后在编译成正常的css文件，以供项目使用

在css中使用 变量、简单的逻辑程序、函数。可以让你的css更加简洁、适应性更强、可读性更佳、更易于代码的维护

146、什么是gulp？作用？机制是什么？

什么是gulp?
基于node的自动化构建工具

作用：
1 自动压缩JS文件
2 自动压缩CSS文件
3 自动合并文件
4 自动编译sass
5 自动压缩图片
6 自动刷新浏览器

机制：
Unix操作系统的管道（pipe）思想 前一级输出 后一级输入