1、event对象---事件对象

事件对象：当触发 onclick, onkeyup ,.........这些事件时，会产生一个事件对象event。

不同事件产生的事件对象对应的属性不同

谷歌浏览器  ： window.event   事件参数   evt

火狐 或  ie ： window.event

事件对象兼容：

     evt || window.event     先兼容高版本浏览器 evt为事件参数

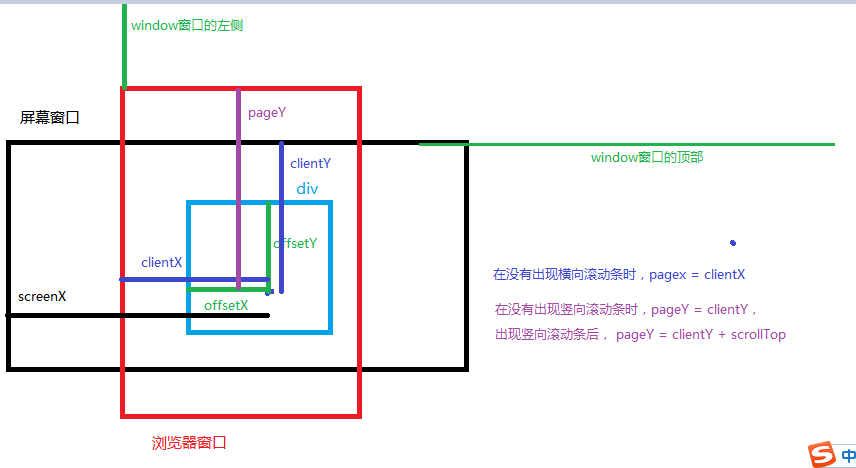
2、鼠标事件对象的坐标属性

clientX  /   clientY   距离 window窗口的  横坐标 和  纵坐标

pageX    /   pageY     距离文档 窗口的  横坐标和 纵坐标   ，没有竖向滚动条时，pageY = clientY   ，出现了竖向滚动条，pageY = clientY + scrollTop

offsetX  /   offsetY   距离目标对象的 内部偏移（左偏移 和 上偏移）

screenX  /   screenY   距离 屏幕 窗口的 横坐标和 纵坐标



3、button属性

判断点击的是 左键  滚轮  右键？

高版本浏览器 ：        左键--0    滚轮 -- 1   右键---2

低版本浏览器 :(IE678)  左键 -- 1    滚轮 -- 4    右键---2

4、鼠标右键菜单 阻止

兼容写法：

     e.preventDefault ? e.preventDefault() : e.returnValue = false;

5、键盘事件对象的属性  keyCode

键盘事件：onkeydown   onkeyup

keyCode属性： 获取键盘的按键值

回车键的keyCode：13

6、事件流

当事件发生时，从子元素向父元素触发 或 从父元素向子元素触发的过程，叫做事件流。

事件流有两种模式：

     事件冒泡 ：  从子元素向父元素触发  small --- big  --- body  --- document --- window

     事件捕获 ：  从父元素向子元素触发  window --- document --- body  --- big  --- small

7、事件冒泡

当事件发生时，同样的事件会再父元素上触发，这个过程就成为事件冒泡

并不是所有的事件都会产生冒泡问题 ： 比如 onload  onfocus  onblur

阻止事件冒泡兼容：

     e.stopPropagation ? e.stopPropagation() : e.cancelBubble = true;

8、事件绑定方法

1、 <input onclick = "fun()">

2、 document.getElementById().onclick = function(){

}

3、 obj.addEventListener("事件",事件处理程序)   注意：这里的事件不加 on

9、事件监听

通过 addEventListener 方式为元素绑定事件成为事件监听

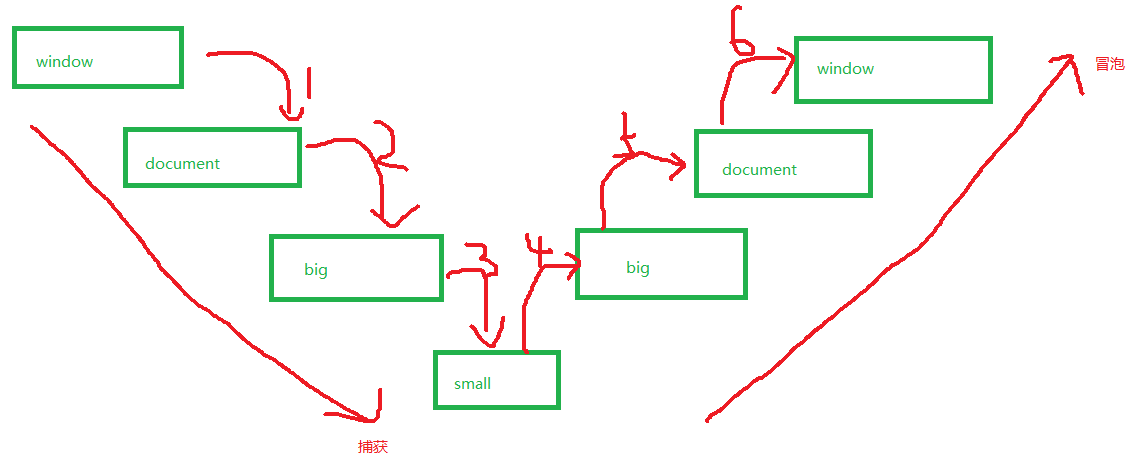
好处： 可以为相同元素 添加 多个同样事件 （执行顺序：从上到下执行）

10、事件捕获

通过事件监听方法 实现事件捕获

addEventListener（事件,事件处理程序,true）    第三个参数 默认是false---冒泡

一个元素既有冒泡 又有捕获先执行谁？？--- 先 捕获  后 冒泡



2、事件委托

委托：     让别人去做

事件委托： 将事件添加到 某些元素的父级元素上，委托这个父级元素来执行这个事件

委托的好处 ：提高程序的性能

          　为动态创建的元素添加事件时，不需要考虑代码书写位置 ，可以写在动态创建的函数体外部

委托实现方式：

     obj.onclick = function(){...}

     obj.addEventlistener("click",function(){})

委托实现的原理：

     利用事件冒泡（捕获也可以）机制，将事件添加到父级元素上 （这里的父级一般是离它最近的直接父级元素）

获取目标元素  ：e.target || e.srcElement

3、事件监听兼容问题

高版本浏览器： addEventListener()    事件不能加on    多个对象添加同样的事件时，从上到下执行 ，有三个参数 ，第三个参数布尔值，true捕获   false冒泡

低版本浏览器： attachEvent()         事件一定加on    多个对象添加同样的事件时，从下到上执行 ，只有两个参数 第一个参数是事件的类型 ， 第二个参数是事件处理程序

4、json对象

什么是json？   javascript object  notation

中文含义：    javascript 对象表示法

json作用：　　存储数据

json定义：   json对象是一种轻量级的数据交换格式。

json格式定义：

    var json = { "":"" }

    严格的json模式  键一定要用 双引号

    json格式由 键值对 组成的 ， 每一个键对应的值 可以是任意类型   多个键值对之间用逗号间隔

    一个json对象中可以有多个键值对

例如：

var json = { "name":"jack", "age" :　20 }   定义时直接赋值

var json = {};     先定义后赋值

json.name = "jack";

json.age = 20

json数据的读取：

     json对象.键

     json对象["键"]

json对象的遍历： 必须用 for... in

5、拖拽

拖拽思路：

     1、拖拽的前提是 鼠标按下左键   这里需要 onmousedown

       鼠标 按下时 ，记录 鼠标相对于目标元素的 内部偏移量 offsetX  和  offsetY

       var disx = e.offsetX;

       var disy = e.offsetY

     2、要想让盒子在鼠标按下时 跟随鼠标移动  需要 给文档添加一个 onmousemove事件（在鼠标按下时添加）

       obj.onmousedown = function(){

          ....

          document.onmousemove = funct(){ ... }

       }

     3、停止拖拽，需要鼠标抬起事件    抬起时  需要停止拖拽  ，取消移动事件  document.onmousemove = null

          obj.onmouseup = function(){

               document.onmousemove = null

          }

获取窗口的宽度和高度：

     window.innerWidth / window.innerHeight   (窗口的内部宽度)

     window.outerWidth / window.outerHeight   ( 包括浏览器的工具栏的高度 )

6、取消拖拽时的文字选中状态

  /\*window.getSelection().removeAllRanges();  高版本浏览器

 document.selection.empty();\*/      低版本浏览器

兼容写法：

 window.getSelection?window.getSelection().removeAllRanges() : document.selection.empty();

2、offset家族属性

offsetWidth/offsetHeight

     用法：　　某元素.offsetWidth   获取结果 ：  内容宽度 + padding + border

clientWidth /  clientHeight

     用法：   某元素.clientWidth   获取结果 ：内容宽度　+ padding

offsetWidth 和  obj.style.width 区别：

1、前者 获取结果类型number   后者结果类型string ，使用时一定要 parseInt

2、前者 可以获取任意样式值   后者只能获取行内样式值

3、前者 只能读数据 ，不能写数据   ，  后者 可读写

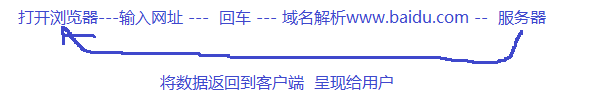
 obj.offsetWidth = 1000  错误

offsetLeft  : 获取某元素中离他最近的具有定位的父级元素的左偏移 ，  如果父级中没有定位的元素，默认相对于body的左偏移

offsetTop   ：  同上  上偏移

3、客户端和服务器

客户端和服务器端工作流程



客户端：  用户  （需要一台电脑访问服务器）

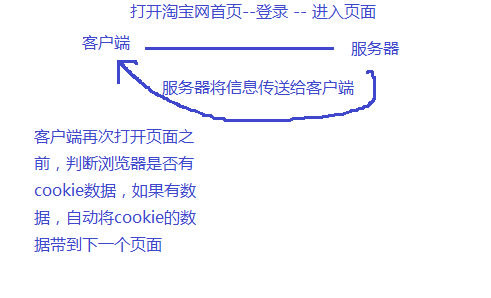
服务器：  为客户端提供服务

         安装指定的服务器软件 apache  --  php        tomcat --  java     iis ---  .net

4、cookie

会话跟踪技术

cookie信息的存储过程 ：



会话： 客户端请求数据，服务器根据客户端请求的数据 将数据返回到页面的这样一个过程 称为一次会话。（如果浏览器被关闭，本次会话结束）

会话跟踪技术 ： 就是cookie

cookie作用 ：存储会话过程中的状态和信息（存储数据）

5、cookie数据的存储

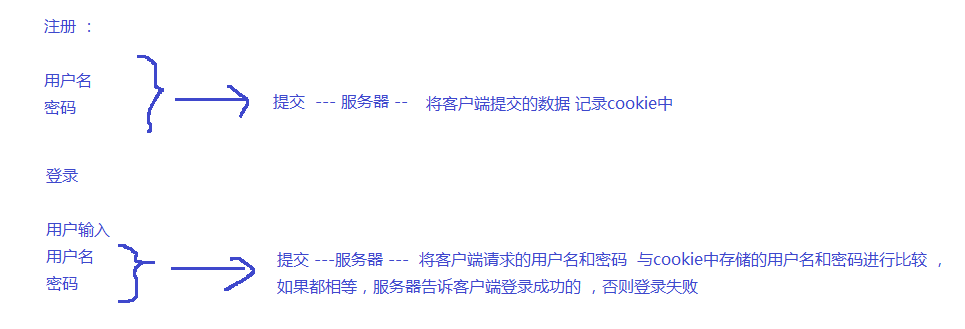
存cookie：

     document.cookie = "键=值"

取cookie：

     document.cookie

cookie实现注册和登录的工作流程：



6、 对象类型和字符串类型转换

从服务器得到的数据都是一个字符串

JSON.parse() 将字符串类型对象转成对象  （将纯对象格式的字符串转成对象）               “abc” 错误

                                        '[{"name":"jack"}]'   '{"name":"jack"}'

JSON.stringify() 将对象转成 字符串

1、cookie生存期

生存期就是cookie在浏览器上保存的时间

生存期设置：   document.cookie = "键=值;expires=生存期"      生存期格式：标准时间格式

定义一个cookie，两天后过期：

     var d = new Date();

     //设置时间为2天后的时间

     d.setDate( d.getDate()  + 2 )

     document.cookie = "name=admin;expires="+d

删除cookie的生存期：

     将cookie值设置为 ""   , 或将生存期设置为  -1

通过更改系统时间的方式删除cookie为假删除 ，系统时间更改后生存期恢复

2、总结cookie

1、cookie不能夸目录访问（不能跨域访问）

2、cookie存储量小  不安全

3、一般cookie 最多存储 50个  ， 大小 4kb

3、正则表达式对象

qq：必须是数字

手机号必须是数字   13..

邮箱 ：   24333@qq.com

正则作用 ： 校验 （前端往往会有大量的工作需要校验，通过正则可以大大的提高工作效率）

4、正则表达式的定义

var reg = /  /修饰符;

修饰符 ：

     i ：　　 不区分大小写

     g ：　　全局查找

     ｍ　：　多行查找

５、正则表达式组成

正则表达式是由普通字符和特殊字符组成的

6、特殊字符

test（）  方法

用法：  reg.test(str)  通过该方法校验str是否符合reg的规范，如果符合，就返回true，否则返回false

特殊字符

\d :  表示 0--9 之间的任意一个数字

\D :  表示 非数字

^ : 表示正则表达式开始

$ : 表示正则表达式的结束

{n}:n是正整数    表示其前面紧挨着的字符只能出现n次

{m,n} : m<n  m n 都是正整数   表示其前面紧挨着的字符至少出现m次  至多出现n次

{m,}   表示其前面紧挨着的字符至少出现m次

\w ： 表示 一个单词字符 （字母或数字或 \_ 中的任意一个字符）

\W :  表示非 字母、数字、下划线中的任意一个字符

[   ] : 表示括号中的任意一个字符

[a-z] : 表示a-z中的任意一个字符

[0-9] 等价  \d

[^a-z] : 非a-z中的任意一个字符

\s :   表示空白字符 （空格、tab）

\S:   表示非空白字符

\* ：  表示 {0,}    表示其前面紧挨着的字符至少出现0次，至多无数

+ :   表示 {1,}     表示其前面紧挨着的字符至少出现1次，至多无数

? :   表示 {0，1}   表示其前面紧挨着的字符至少出现0次，至多1次

|  :   表示或者    一般和() 连用

.  :   表示任意一个字符

\  :   转义字符           一般用来对正则中的 特殊单个字符进行转意    \\  ----\        \.-----.

[\u4e00-\u9fa5]:  表示任意一个中文字符

正则表达式方法

test() 检验一个字符串是否符合某一个正则规范，如果符合返回true，否则返回false

用法： reg.test(str)

exec()  根据正则表达式查找，结果会返回一个长度为1的数组 （数组只有一个值）  找不到返回null

用法： reg.exec(str);

search方法     返回与正则表达式查找内容匹配的第一个子字符串的位置。

 用法：  str.search(reg);

match() 方法   使用正则表达式模式对字符串执行查找，并将包含查找的结果作为数组返回。

 用法：  str.match(reg)

 说明：    正则如果有g  表示全局匹配，返回一个数组

          正则如果没有g   表示match只能在str中匹配一次，返回的第一个元素是匹配的正则结果，其余的元素是  正则中子表达式匹配的结果

str.replace(原字符串,新字符串)

replace（）   通过正则表达式替换，返回结果 是原字符串被替换后的新字符串

用法：  str.replace(正则表达式,要替换的新字符串)      如果想实现全局替换，在正则表达式上加一个修饰符 g

ES5中严格模式下的要求

严格模式： "use strict"

严格模式下的要求：

              不加var报错

              八进制被禁用（不能用0开头）

              function必须声明在顶层

ES6

 数组新增方法---  forEach    reduce   filter   map    indexOf

 bind 方法  绑定匿名函数中的this指向

 apply方法 和 call方法

     function fun(){

          console.log( this );

     }

     fun.apply(document);//即改变了this指向 又调用了 fun函数  让fun函数中的this 指向了 document

常量const　常量的值定义后就不能再更改

     为了和普通变量区分开，常量定义时使用大写字母

     常量一旦被定义，值就不能被更改

     const PAI =  3.14; //定义了一个常量

     alert( PAI\*2\*2 );

for...of循环  和  for  ... in 循环：前者遍历值 ， 后者遍历 键

let块级作用域 ：

     let 不可以实现变量提升

     let 只在 {} 括号内是有效的   作用在块级作用域{}  上