**Javascript**

# 一、注释

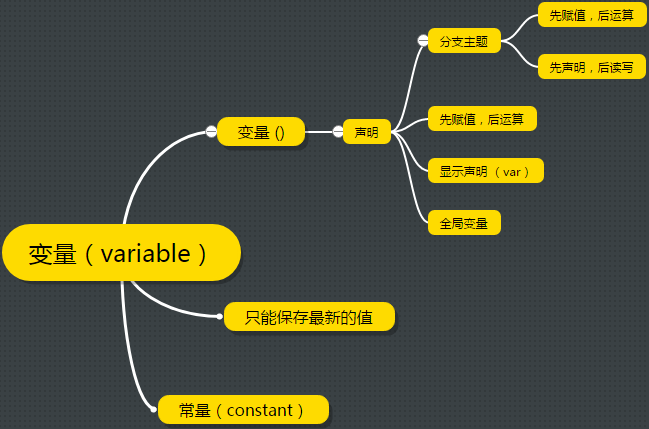
1. 单行//
2. 多行/\*\*/
3. 块级/\*\* \*/

# 二、浏览器历史

1. netscape-->navigator
2. 宿主：浏览器
3. livescript-->javascript
4. internet exporer
5. jscript-->vbscript
6. 2005
7. XMLHttpRequest-->后台
8. ajax
9. google --》gmail,google map
10. nodeJS
11. andriod->java
12. ios:obejct-c;
13. js+php
14. jsp aspx php

# 三、javascript

* + Javascript是弱类型的脚本语言
  + 变量
  + 声明变量
  + var 显式声明，
  + 如果没有var的话，是全局变量。
  + 变量只能保存最新的值。
  + variable:变量。
  + const：constant常量。



## 字符串

* 1. 引号成对出现
  2. 不能嵌套自身
  3. 示例，将img标签src中的地址替换为num变量
     1. “<img src='"+num+".jpg'>”
     2. '<img src="'++'.jpg">'

## 字符串方法

* 1. charAt();
  2. indexOf()
  3. lastIndex()
  4. concat()
  5. slice();
  6. substring();

## 数组

### 数组概念

* + 1. 数组可以保存多个值
    2. 创建数组,最后一个元素的后面，不能加逗号。
    3. 读取数组的元素member[index]
    4. var member=[];空数组
    5. new Array(5):数组的长度是5；
    6. var arr= new Array()声明数组

### 数组方法

1. push()
2. pop();
3. unshift()
4. shift();

## 流程控制

1. **条件**
   1. **If条件语句**
2. if(expr){//code}
3. if(expr)else{//code}
4. if(expr){//code}else if(){}
   1. **switch条件语句**

switch(value){

        case :

        ///code

        break;

}

### c) 三元条件

结构

表达式？结果1：结果2；

如果表达式为真，就执行结果1，

如果为假，就执行结果2。

## 循环

**For循环**

for(expre1;expre2;expre3)

for(var i=0;i<5;i++){

console.log(i);

}

---------------------------------------------------------------------------------

**i++  i=i+1；**

**i+=2;i=i+2;**

---------------------------------------------------------------------------------

* 1. for...in,只针对数组和对象

for(i in member){}

i就是member的索引值

* 1. for...of

for(i of member){}

i指代的是数组的元素。

* 1. forEach(function(value,index,arr){})循环

通过回调函数传递数组元素，

索引值，数组本身。

member.forEach(function(value

，index，arr){ });-

--------------------------------------------------------------------------------

**while循环**

* 1. while(expre){//code}
  2. do{//code}while(expre)就算表达式为假，循环也会执行一次。
  3. 跳出break,continue跳出正常的代码执行逻辑

**break**：终止代码执行。

**continue:**忽略这次循环，继续执行。

--------------------------------------------------------------------------------

**内置对象**

1. **时间对象，处理的本地时间。**

Date是js中原生的时间对象，需要实例化才可以使用。

1.**new Date()**;获取输出本地时间的对象

2.**new Date(2018,09,12)**;指定的时间对象

1. **时间对象的方法**

* getFullYear();年份
* getMonth()+1:月份 0~1
* getDate();日
* getHours():时
* getMinutes()：分
* getSeconds()：秒
* getDay()星期0~6
* getTime();时间戳timestamp,单位是毫秒，int类型；

**元素.innerHTML**；既可以为元素赋值，也可以取值，支持html。

setInterval(code,时间);设置时间间隔（在每隔设定的时间就执行一次code）

|  |
| --- |
| **0517作业**  1.基础知识，不会的mark  2.图片时间静止  3.图片时间动态 |

----------------------------------------------------------------------------------------------------

# 数学对象(Math())

1. **属性**

console.log(Math.PI);圆周率

1. **方法**
   1. console.log(Math);
   2. console.log(Math.min(5,14));小值
   3. console.log(Math.max(5,14));大值
   4. console.log(Math.pow(2,4));幂
   5. console.log(Math.floor(2.9));最近的小值
   6. console.log(Math.ceil(2.9));最近的大值
   7. console.log(Math.random());随机数0~1
   8. console.log(Math.round());四舍五入
2. **字符串对象**
3. **创建字符串对象**

new String();

1. **属性**
   * length字符串的长度
   * 字符串可以根据索引值获取字符
   * str[index]
2. **方法**
3. substring(start,end);截取字符串
   * + start:开始截取的索引值
     + end：结束截取的索引值，结果不包含end。
4. slice(start,end)
   * + 截取字符串，支持负值。
5. split(分隔符)根据分隔符把字符串转换为数组
   * + 如果符串中没有分隔符的话，整个符串作为数组的一个元素。
     + 如果分隔符为空，字符串中的所有字符都是数组中个一个元素。

|  |
| --- |
| **学习技能**  **https://develo per.mozilla.org/zh-CN/**  **http://stackoverflow.com/** |

1. **数据类型**

**转换：自动转换和函数转换**

示例：转换为数字

|  |  |
| --- | --- |
| 类型 | 值类型 |
| Number()（数值型） | 55 |
| Boolean()（布尔型） | “true”“false” |
| parseInt()（整数型） | "55ab"==>55 |
| parseFloat()（浮点型） | "5.5abc.com"==>5.5 |

Number():"55"

Boolean()

parseInt():"55ab"

parseFloat():"5.5abc.com"==>5.5

1. **函数，function函数，****argument（实参）参数** parameter（**形参**）

封装可复用的代码

1. **结构**
   * function 函数名(参数列表){

//函数体，封装的可复用的代码

}

* + 参数列表：参数可以为空，参数有顺序要求。

1. **函数需要触发**

* 直接调用函数名
* 通过元素的事件属性调用

<button onclick="Msg();">按钮</button>

* 事件

绑定事件:元素.addEventListener()

1. **函数的返回值**

return是关键字，return之后的内容不执行

* 字符串

return “tom"

* 数字
* 布尔型
* 数组

return ["张三","李四"];

* 对象

 return {

                "username":"tom",

                "gender":"male",

                'level':"vip"

        }

* 函数

    return function(){

                alert(555);

       }

1. **函数的参数**

  function Msg(username){

        return username;

}

username:形式参数，形参。

Msg("aaaa");实际参数，实参。

1. **匿名函数**

  function(){}

1. **匿名函数的调用**

* 赋给变量

var Msg=function(){

                alert(333);

        }

* (匿名函数)()匿名函数的自触发

1. **匿名函数的传值**
2. 类型一 **()()**

(function(username){

                alert(username);

        })("tom");

1. 类型二 **实参传入形参**

var Msg=function(username){

alert(username);

} Msg("peter");

1. **箭头函数**

|  |  |
| --- | --- |
| **var Msg = value => value** | function Msg(value){         return value;    } |

1. **Function构造方法创建函数**
   * var Msg=new Function("return 'tom'");
   * var Msg=new Function("username","return username");
2. **arguments(实参)**
   * 数组类型，包含的是实参。

|  |
| --- |
| function Msg(username,level){          console.log(arguments);  }  Msg("mike","vip"); |

# 对象

  {key:value,key:value,key:value}

**元素居中**

|  |
| --- |
| 1..demo{                  width:300px;                  height:100px;                  border:1px solid red;                  position:absolute;                  left:50%;                  top:50%;                  margin-left:-150px;                  margin-top:-50px;          } |
| 2..demo{                  width:300px;                  height:100px;                  border:1px solid red;                  position:absolute;                  left:0;                  right:0;                  top:0;                  bottom:0;                  margin:auto;          } |
| 3.js        （窗口的尺寸-元素的尺寸）/2+滚动条的值 |
| **0518作业**  1.预习**DOM文档对象模型**  **2.不同时段的欢迎语句**  **3.封装根据类名称获取元素** |

# DOM文档对象模型

* **DOM**：document object modal
* 把文档看成一个对象，根据文档的层级操作文档。
* **DOM Tree**
* 文档树由节点组成，节点是文档的最基本单位。
* 节点：node

1. **节点的类型 nodeType**

* 元素节点    1
* 属性节点    2
* 文本节点    3
* cdata节点
* 实体引用节点
* 实体节点
* 处理指令节点
* 注释节点
* 文档节点
* 文档类型节点
* 部分文档节点
* DTD声明符号节点

1. **节点的属性**

* nodeType：节点类型
* nodeValue：节点值
* nodeName：节点名称
* firstChild：第一个子节点
* lastChild:最后一个子节点
* childNodes:子节点数组
* parentNode：父节点
* previousElementSibling：同级元素紧邻前面一个
* nextElementSibling：同级元素紧邻后面一个

1. **节点的的方法**

* appendChild():追加子节点
* createTextNode();创建文本节点
* removeChild();移除子元素

**node中如何清除空格**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | <!DOCTYPE html> | |  | <html> | |  | <head> | |  | <title></title> | |  | </head> | |  | <body> | |  | <ul class="demo"> | |  | <li>one</li> | |  | <li>two</li> | |  | <li>three</li> | |  | <li>four</li> | |  | </ul> | |  | <button class="btn">Button</button> | |  | <script> | |  | var demo=document.querySelector(".demo") | |  | var btn=document.querySelector(".btn") | |  | btn.addEventListener("click",function(){//alert(demo.firstChild.nodeType); | |  |  | |  | if(demo.firstChild.nodeValue!="#text"){ | |  | demo.removeChild(demo.firstChild); | |  | } | |  |  | |  | }); | |  | </script> | |  | </body> | |  | </html> | |

# 元素(element)

1. **选择元素**
2. getElementById(id名称)，根据id名称选择元素，返回值是获取到的这个元素，如果获取不到，返回null。
3. getElementsByTagName(),根据标签名选择元素，返回值是包含选择元素的数组，如果获取不到，返回空数组。
4. getElementsByClassName();根据类名称选择元素，返回值是包含选择元素的数组，如果获取不到，返回空数组。
5. getElementsByName();根据name选择元素，返回值是包含选择元素的数组，如果获取不到，返回空数组。
6. querySelector();根据选择器获取元素，返回值是符合要求第一个元素，如果没有结果，返回null。
7. querySelectorAll()；根据选择器获取元素，返回值是符合要求所有元素，如果没有结果，返回空数组。
8. **属性**

* 属性在元素的开标签中，属性名之间，属性名与标签名之间至少有一个空格。
* 合法和自定义属性。
* 属性值既可以获取，也可以赋值。

1. **合法属性**

* class要变为className才可以处理元素的class属性。
* sytle样式属性

如果样式是有多个单词组成，单词链接在一起，除了第一个单词外，后面单词的首字母大写。demo.style.fontSize

## 自定义属性

* **方法**
* appendChild():追加子节点

1. **匿名类，伪类**

|  |
| --- |
| hover,link,visited,active,after,before,target,focus,blur,checked,dropback,fullscreen, |

|  |
| --- |
| **0519作业**  换皮肤  预习事件 |

1. **link标签**，链接外部的文件。

* rel：relationship关系。stylesheet样式表，shortcut：快捷图标
* <link rel="stylesheet" type="text/css" href="a.css">
* <link rel="shortcut" href="a.ico">

1. **setInterval(code,delay);**

* 设置时间间隔，每隔delay的时间，就执行一次code。
* code：两种类型，string，function。
* delay：间隔时间，单位是毫秒。

1. **元素的属性**

* **既可以取值，也可以赋值。**
* **合法**

|  |
| --- |
| className,id,size,width,height,title,href,color,bgcolor,style,checked,selected,name,  type,for,placeholder,value, |

* **value表单元素的值属性**
* Input元素中的value是收集的用户的数据。
* **innerHTML普通元素的值**
* **获取元素的属性**

1. **clientWidth**:元素的宽，数字类型， 包含内容和padding。
2. **clientHeight**:元素的高，数字类型， 包含内容和padding。
3. **offsetWidth**:包含边框，内容，内边距，滚动条。
4. **offsetHeight**:包含边框，内容，内边距，滚动条。
5. innerWidth:
6. innerHeight:
7. **scrollWidth**:滚动条的宽
8. **scrollHeight**:滚动条高
9. **scrollLeft**:水平滚动条滚动的值
10. **scrollTop**:垂直滚动条条滚动的距离

* **窗口的宽度**
* window.innerWidth：Chrome,ie不起作用。
* document.documentElement.clientWidth：ie，chrome不起作用。

**文档对象**

document

**HTML对象**

document.documentElement

netscape网景---》AOL--》Mozilla--》firefox 20%

navigator浏览器

chrome

ie6-->ie7--->ie8->ie9-11  edge

1. **自定义属性**

|  |
| --- |
| setAttribute(key,value)设置属性值，合法和自定义都可以 |
| getAttribute(key)获取属性值，合法和自定义都可以。 |

1. dataset统一处理自定义属性

|  |
| --- |
| 自定义属性名以data-开头，在js中用dataset获取。 |
| <h1 class="demo" data-url="http://www.qq.com" data-info="hello">hello</h1>  <script type="text/javascript">          var demo=document.querySelector(".demo");          console.log(demo.dataset.url);          console.log(demo.dataset.info);  </script> |

|  |  |  |
| --- | --- | --- |
| **属性** | **合法** | **自定义属性** |
| setAttribute(key,value),设置属性  getAttribute（key）,获取属性  removeAttribute()，移除属性  可以处理合法和自定义属性。 | 元素.属性名 | * setAttribute(),getAttribute（） * data-\*在元素中命名，在js中处理元素.dataset.自定义属性名 |

|  |
| --- |
| **0522作业**  登录界面 html，css，js  复习居中函数 |

**innerHTML：普通元素的值**

**value：表单元素的值**

**10 页面内的导航**

|  |
| --- |
| http://www.163.com:8080/reg/index.php#top?a=action&page=5 |
| **协议：//host:port/path/index.php#section?searchQuery(查询字符串)** |
| 1.a name="top"  2.元素  id="top" target目标选择器  #top:target{           color:red;           font-size:100px;   } |

 3.js

 btn.addEventListener("click",function(){

       document.documentElement.scrollTop=0;

       document.body.scrollTop=0;

});

# 事件

* Event页面上的行为就是事件。

1. **元素绑定事件**

元素.on事件名称=匿名函数

img.onclick=function(){

       //console.log(this);

       this.style.display="none";

}

//////////////

img.onclick=disappear;

function disappear(){

        this.style.display="none";

}

1. **元素.addEventListener();调用元素的添加事件侦听器方法，调用绑定事件的方法。**

* 特点：一个元素可以绑定多个事件，一个事件可以多次使用。

元素.addEventListener(type,listener);

type:事件类型

listener：事件侦听器，也就是事件的内容。

html.addEventListener("click",function(){});

//////////////////////////////

html.addEventListener("click",msg);

 function msg(){

                console.log("hello");

}

**0523作业**

点击按钮图片显示，再点击按钮，图片显示。（多个版本）

|  |
| --- |
| **1.显示隐藏**  btn.addEventListener("click",function(){                  if(img.style.display=="none"){                          img.style.display="block";                  }else{                          img.style.display="none";                  }  }); |
| **2.三元**  img.style.display=(img.style.display=="none")?"block":"none"; |
| **3.布尔型变量的切换**  var b=true;          btn.addEventListener("click",function(){                  if(b){                          b=false;                          img.style.display="none";                  }else{                          b=true;                          img.style.display="block";                  }                  console.log(b);          }); |
| **4.switch**  switch(img.style.display){                          case "":                                  img.style.display="none";                                  break;                          case "none":                                  img.style.display="";                                  break;  } |

**Tab页效果**

|  |
| --- |
| **1.forEach**  lis.forEach(function(value,index){                  value.addEventListener("click",function(){                          //第一个步清空所有元素的active                          lis.forEach(function(v,i){                                  v.className="";                                  divs[i].className="";                          });                          //console.log(index);                          //为当前的元素添加active;                          value.className="active";                          divs[index].className="spotlight";                  });          }); |
| **2.setAttribute,getAttribute()**  for(var i=0;i<lis.length;i++){                  //循环为每个一个list添加index                  lis[i].setAttribute("index",i);                  lis[i].addEventListener("click",function(){                          for(var j=0;j<lis.length;j++){                                  lis[j].className="";                                  divs[j].className="";                          }                          //console.log(this.title);                          divs[this.getAttribute("index")].className="spotlight";                          this.className="active";                  });          } |
| **3.匿名函数的自触发**    for(var i=0;i<lis.length;i++){                  //循环为每个一个list添加index                  //匿名函数的自触发  (形参)(实参)                          (function(index){                                  lis[index].addEventListener("click",function(){                                          for(var j=0;j<lis.length;j++){                                                  lis[j].className="";                                                  divs[j].className="";                                          }                                          //console.log(this.title);                                          divs[index].className="spotlight";                                          this.className="active";                                  });                          })(i);          } |

1. **单击事件(click)**

  tab页

  全选

  双击事件(dblclick)

1. **鼠标事件**
   * mouseover
   * mouseout

|  |  |
| --- | --- |
| **0524作业**  Tab页 鼠标事件   动画效果的导航 | **PK**  居中  Tab页 |

1. **change事件**

* 每次选择的值有不同时才会触发。
* 一般用在select或者input type=“file"上

1. **焦点事件**

* focus:获取焦点
* blur:失去焦点

1. **按键事件**

* keyup：松开按键
* keydown：按下按键

1. **copy和paste**

var demo=document.querySelector(".demo");

|  |  |
| --- | --- |
| * copy：复制 | demo.addEventListener("copy",function(){                  alert("copy");          }); |
| * paste：粘贴 | demo.addEventListener("paste",function(){                  alert("paste");          }); |
| * cut：剪切 | demo.addEventListener("cut",function(){                  alert("cut");          }); |

1. **事件对象**

* 事件触发时，事件对象产生，事件结束时，事件对象消失。
* 开发者使用事件对象的属性和方法进行操作。

1. **创建事件对象**

* 事件侦听器中的形参：chrome，firefox，
* 在IE中，事件对象是window对象的子对象 var evt=window.Event;

1. **方法**

* stopPropagation()：阻止事件传递（ie下有差异）
* preventDefault();阻止默认动作（ie下有差异）

1. **属性**

keyCode:键值

1. **前端动画**

svg，canvas+js，css，js，jquery

flash，Actionscript

o'rellily 禽兽

apress    衣冠

1. **加载事件**
2. **load**

window.addEventListener("load",fn)，页面元素都加载完成。

1. **DOMContentLoaded**

DOMContentLoaded:DOM内容加载完成。<img src="1.jpg">

load：所有元素。<img src="1.jpg">

1. script标签的defer属性
2. **卸载事件**

window.addEventListener("unload",fn)

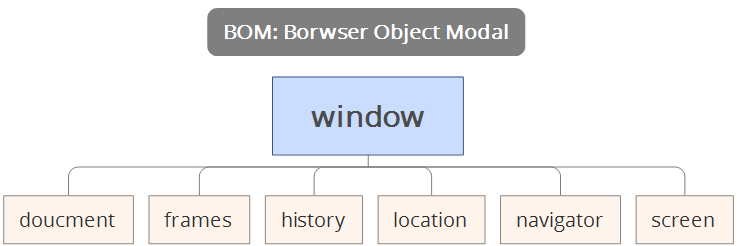
1. **同步**

按顺序执行

1. **异步**

异步代码不在顺序执行的序列中。

# BOM

* **window对象是最上层对象，下面有documet，frames，history，location，navigator，screen对象**

1. **window**
2. **属性**

name：窗口名称

1. **方法**
   1. **open**（url，name,features）:打开新窗口。
      * + url:要打开的文件的路径。
        + name：打开窗口的名称。
        + features：特性字符串，打开窗口的坐标，样式等。
        + 浏览器默认拦截
   2. **close()**关闭浏览器
   3. window.setInterval(code,delay)时间间隔
   4. window.setTimeout(code,delay)：delay结束时，code执行一次。
      * + setInterval和setTimeout返回值是定时器的id
        + clearTimeout,clearInterval
   5. moveTo:浏览器移动到
   6. moveBy：浏览器移动多少
   7. resizeTo：大小改变到
   8. resizeBy：大小改变了多少
   9. alert:数据信息不如console.log详细，需要用户干预才能向下走。
   10. confirm(提示信息):确认框，返回true或者false
   11. prompt(提示信息，初始值)：输入框，初始值可以为空。
2. **事件**
   * + load,unload,resize调整大小,scroll,
3. **document对象**

子对象

images,links,anchors

**对象的层级**

window.document.images[1].style

1. **location对象**
2. **地址栏对象**
   1. **属性**

* location.href;URL
* location.port;端口
* location.pathname;网站路径
* location.host;主机名
* location.protocol:协议file,
* location.search:查询字符串?a=show&page=1
  1. **方法**

reload(）重新加载

1. **history历史记录对象**

**方法**

* history.back()后退一个历史记录
* history.forward()前进一个历史记录
* history.go(n):n为负数是后退，n为正数，是前进。

1. **screen屏幕对象**

* width:宽度
* availWidth：可用宽度
* height:高度
* availHeight：可用高度

1. **navigator 浏览器对象**

appVersion版本

# 正则表达式

regular expression

**1. 创建正则对象**

1) new RegExp(模式,模式修饰符);

var pattern=new RegExp("sh",'i')

1. 对象字面量的形式创建对象

pattern=/sh/

1. **匹配**

字符串中有符合模式的内容，就称为匹配。模式的字符有顺序要求。

1. **正则函数**

* test: pattern.test(str):检测pattern在str中是否有匹配的结果，返回值是布尔型。
* exec: pattern.exec(str);返回值是数组，里面包含了符合的第一个结果。如果没有匹配，返回null。
* match：str.match(pattern);返回值是数组，里面包含所有的匹配的结果，如果没有，返回null。
* search：str.search(pattern);返回匹配的字符出现的第一个位置的索引值，如果没有，返回-1。(类似于indexOf)

1. **模式修饰符**

* **g** : global全局匹配
* **i**：insensitive大小写不敏感
* **m** : 多行
* **^** : 在行首查找
* **$** : 在行尾查找
* 禁止贪婪：.+**?**

1. **元字符**

* [ ] : 表示一个字符
* [ab] : a，b只要有一个匹配就为true
* [^a]：排除a
* [a-z]：任意一个小写字母
* [A-Z]：任意一个大写写字母
* [0-9]：任意一个数字=====[\d],[\D]:非数字。
* [0-9a-zA-Z] :
* [A-Za-z\_]====[\w],[\W] : 非[A-Za-z\_]
* . : 匹配非空任意字符
* \. : 转义为普通的点
* \s : 空格
* \S : 非空格

1. **量词，修饰前导字符**

* ？：0次或者1次。
* \*：任意次，包括0次。
* +：至少一次。
* {n}:n次
* {n,}:至少n次
* {n,m},最少n，最多m。

1. **分组**

* 捕获性分组

1. **常用正则**

1. ubb转换为html

2. 验证邮编   /^[1-9][\d]{5}$/

3. 验证密码强度

4. 验证邮箱

|  |
| --- |
| **端午作业**  1.输入为空时，把空格删除  需要处理的trim中全是空格的情况。  trim(" ");  function trim(value){          var pattern=/^(\s\*)$/;          if(pattern.test(value)){                    value=value.replace(pattern,"$1");                  console.log(value.length);          }else{          }    }  2.复习JavaScript  3.预习ajax |

**Ajax**

1. **网络环境**

* Apache
* IIS:internet information service

1. **后台语言**

* PHP,7,面向对象
* jsp
* aspx

1. **数据库**

* mysql+phpmyadmin

[127.0.0.1/phpmyadmin]

sql server:ms

oracle:

1. **ajax对象**

* Asynchrous JavaScript And  XML
* Asynchrous JavaScript And  PHP+MySQL

1. **属性**

* xhr.responseText:以字符串的形式接收后台返回的数据。
* xhr.responseXML:以XML的形式接收后台返回的数据。
* xhr.readyState:状态值,0,1,2,3,4,4是交互成功。
* xhr.status:http的状态码，404文件未找到，200：返回数据成功。

1. **方法**

* xhr.open("post","reg.php"):xhr链接php文件
* xhr.send();向PHP文件发送数据

1. **事件**

* readystatechange状态改变事件