学习方法

1. 建立逻辑思维
   1. 多分析
   2. 理解逻辑算法
   3. 多练、多敲、多读
2. 编程习惯
3. 大胆动手
4. 保持好奇好
5. 乐观面对错误

就业能力

1. 分析能力
2. 独立思考能力
3. 程序阅读能力
4. 程序排错能力
5. 独立编写程序能力
6. 团队协作能力
7. 学习能力
8. 善于总结的能力

专业技能

设计模式及JQ入门

1. 单例模式
   1. 概念：单个实例，只有一个对象；多次创建，返回同一个对象
      1. function Singleton(){  
          if(!Singleton.instance){  
          Singleton.instance = {  
          name : "zzl"  
          };  
          }  
          return Singleton.instance;  
          }  
          var obj1 = new Singleton();  
          var obj2 = new Singleton();  
          alert(obj1 == obj2);
      2. var obj = {

name : "zhangsan",

showName : function(){

return this.name;

}

}

1. 代理模式
   1. 代理模式的作用，在于执行目标函数之前，可以先做其他的相关操作。
   2. 在下面的例子中，地图服务就是一个代理，它跟目标函数一样提供计算航线的功能，但是，它使用了缓存技术来提供数据。这样做的目的很简单，就是在不修改目标函数的前提下，增强功能。
      1. function 计算航线（港口1，港口2）{  
          //ajax()…..  
         }  
         function 地图服务（）{  
          var缓存 = {};  
          this.计算航线 = function(港口1,港口2){  
          //缓存中有内容，则使用缓存，不再计算  
          if(缓存[港口1+港口2]){  
          return 缓存[港口1+港口2];  
          }else{  
          var 某一条航线 = 计算航线（港口1,港口2）;  
          //每计算一次，将结果放入缓存  
          缓存[港口1+港口2] = 某一条航线;  
          return 某一条航线;  
          }  
          }  
         }  
         new 地图服务.计算航线（）;
2. 适配器模式
   1. 我们要对所有产品都进行一个标准化的测试，测试流程其中包括了电话、短信、游戏、音乐等等功能，但是对于一个平板来说，电话功能是无法使用的，因此测试会出问题。所以将平板进行了包装（类似于代理），这样就简单的解决了兼容问题。
      1. function test(product){  
          try{  
          product.phonecall();  
          }catch(e){  
          console.log("电话功能测试失败！");  
          }  
          try{  
          product.playgame();  
          }catch(e){  
          console.log("游戏功能测试失败！");  
          }  
         }  
         function Phone(){  
          this.phonecall = function(){  
           
          }  
          this.playgame = function(){  
           
          }  
          this.toStrina = function(){  
          return "电话";  
          }  
          }  
          function Pad(){  
          this.playgame = function(){  
           
          }  
          this.toStrina = function(){  
          return "平板";  
          }  
          }
   2. 适配器的意义，多数应用在系统接口使用，也就是别人提供了一个功能，但要求传入一个A类型的参数，而我们手里的数据是B类型，如果我们想使用这个功能，那么有两种解决办法：第一，把自己的原代码进行修改，让B类型改为A类型，这里非常蠢的做法。第二，我们把B类型的数据进行一个包装，让它看起来符合类型A，这个包装函数，就是适配器。
      1. function Adapter(product){  
          this.phonecall = function(){  
          if(product.phonecall){  
          product.phonecall();  
          }else{  
          console.log("this" + product.toString() + "is not support function phonecall!");  
          }  
          }  
          this.playgame = function(){  
          if(product.playgame){  
          product.playgame();  
          }  
          }  
          }  
          test(new Phone());  
          test(new Adapter(new Pad()));

第二节：应用

1. 使用单例模式封装信息提示框插件

第三节

1. 选择器
   1. jQuery是什么：是一个javascript代码仓库，我们称之为javascript框架。  
       是一个快速的简洁的javascript框架，可以简化查询DOM对象、处理事件、制作动画、处理Ajax交互过程。
   2. jQuery优势
      1. 体积小，使用灵巧(只需引入一个js文件)
      2. 方便的选择页面元素(模仿CSS选择器更精确、灵活)
      3. 动态更改页面样式/页面内容(操作DOM，动态添加、移除样式)
      4. 控制响应事件(动态添加响应事件)
      5. 提供基本网页特效(提供已封装的网页特效方法)
      6. 快速实现通信(ajax)
      7. 易扩展、插件丰富
   3. 如何下载JQuery
      1. 官方网站：http://jquery.com/
   4. 如何引入JQuery包
      1. 引入本地的Jquery
      2. 使用Google提供的API导入   
         <script type=“text/javascript” src=“jquery.js”></script>  
          写第一个JQUery案例  
          <script type=“text/javascript” src=“”></script>  
          <script type=“text/javascript”>  
          $(function(){  
          alert(“jQuery 你好!”);  
          });  
          </script>
         1. 在JQuery库中，$是JQuery的别名，$()等效于就jQuery().
         2. $是jQuery别名。如$()也可jQuery()这样写,相当于页面初始化函数，当页面加载完毕，会执行jQuery()。
         3. window.onload == $(function(){})
   5. 选择器（讲API）
      1. jQuery选择器分为：基本选择器 ；   
          层级选择器 ；   
          常用伪类选择器:可以看作是一种特殊的类选择符;
      2. jQuery基本选择器
         1. 包括ID选择器，标签选择器，类选择器，通配选择器和组选择器5种
            1. ID选择器：$(“#id”)
            2. 标签选择器：$(“element”)
            3. 类选择器：$(“.className”)
            4. 通配选择器：$(“\*”)匹配指定上下文中所有元素
            5. 组选择器：$(“selector1,selector2,selectorN”)特点：无数量限制，以逗号分隔（逐个匹配，结果全部返回）
         2. 案例  
            <div id="main">孔子</div>  
            <h4>论语</h4>  
            <div class="one">子在川上曰：</div>  
            <p>"逝者如斯夫！</p>  
            <p>不舍昼夜。"</p>
            1. 找到.one改变他的字体颜色
            2. 找到#main给他增加border：1px solid red
            3. 找到p标签元素给他添加边框border:1px solid green
            4. 找到全部元素改变字体28px
            5. 找到ID ClassName 添加background:red
      3. 层级选择器：通过DOM的嵌套关系匹配元素
         1. jQuery层级选择器：包含选择器、子选择器、相邻选择器、兄弟选择器4 种
            1. 包含选择器：$(“a b”)在给定的祖先元素下匹配所有后代元素（不受层级限制）
            2. 子选择器：$(“parent>child”)在给定的父元素下匹配所有子元素。
            3. 相邻选择器：$(“prev + next”)匹配所有紧接在prev元素后的next元素。
            4. 兄弟选择器：$(“prev ~ siblings”)匹配prev元素之后的所有sibling元素。
         2. 案例  
            <div class="main">  
             <span>1<img src="img/1.jpg"/></span>  
             2<img src="img/1.jpg"/>  
            </div>  
            3<img src="img/1.jpg">  
            4<img src="img/1.jpg">  
            <div>  
             5<img src="img/1.jpg">  
            </div>  
            <script type="text/javascript" src="js/jquery.js"></script>  
            <script type="text/javascript">  
             $(function(){  
             //$(".main img").css("border","5px solid red");  
             //$(".main > img").css("border","5px solid blue");  
             //$(".main + img").css("border","5px solid blue");  
             //$(".main ~ img").css("border","5px solid blue");  
             });  
            </script>
         3. 综合应用  
            <h1>沁园春·雪</h1>  
            <h2>毛泽东</h2>  
            <div>  
             <span><div>北国风光，千里冰封，万里雪飘。  
             <div>望长城内外，惟余莽莽；大河上下，顿失滔滔。</div>  
             <p>山舞银蛇，原驰蜡象，欲与天公试比高。</p>  
             <p>须晴日，看红装素裹，分外妖娆。</p>  
             </div></span>  
             <p id="mp">江山如此多娇，引无数英雄竞折腰。</p></div>  
             <p class="c1">惜秦皇汉武，略输文采；唐宗宋祖，稍逊风骚。</p>  
             <p class="c1">一代天骄，成吉思汗，只识弯弓射大雕。</p>  
             <p>俱往矣，数风流人物，还看今朝。</p>
            1. 让id为mp的元素文字大小变成30px
            2. 让class名为.c1的元素背景为#CCFF99
            3. 将所有的div和p，统一显示间距
            4. 将所有的div增加 2像素 实线 红色
            5. 将div下的所有子div中文字变蓝色blue
            6. 将div中包含的div颜色变成#FF99FF
            7. 将div的所有兄弟标签p的字体颜色变成蓝色
            8. 将紧跟着div的p标签的边框设为2像素 实线 #009900
      4. 常用伪类选择器：可以看作是一种特殊的类选择符  
         选择器 说明  
         :first 匹配找到的第1个元素  
         :last 匹配找到的最后一个元素  
         :eq 匹配一个给定索引值的元素  
         :even 匹配所有索引值为偶数的元素  
         :odd 匹配所有索引值为奇数的元素  
         :gt(index) 匹配所有大于给定索引值的元素  
         :lt(index) 匹配所有小于给定索引值的元素  
         :not 去除所有与给定选择器匹配的元素
         1. 案例  
            <ul>  
             <li>1</li>  
             <li>2</li>  
             <li>3</li>  
             <li>4</li>  
             <li>5</li>  
             <li>6</li>  
             <li>7</li>  
             <li>8</li>  
             <li>9</li>  
             <li>0</li>  
             <li>1</li>  
             <li>2</li>  
             <li>3</li>  
             <li>4</li>  
             <li>5</li>  
             <li>6</li>  
             <li>7</li>  
             <li>8</li>  
             <li>9</li>  
            </ul>  
            $(function(){   
             //$("ul li:first").css("border","1px solid red")  
             //$("ul li:last").css("border","1px solid red")  
             //$("ul li:lt(3)").css("border","1px solid red")  
             //$("ul li:not(li:eq(5))").css("border","1px solid red")  
            });
2. 属性(讲API)
   1. 案例  
      <style type="text/css">

ul,ol {

list-style: none;

margin: 0;

padding: 0;

}

ul{

width: 500px;

height: 30px;

}

ul li {

height: 30px;

padding: 0 20px;

float: left;

cursor: pointer;

}

ul li.normal {

color: olive;

background: #abcdef;

border: 1px solid olive;

border-bottom: 0;

}

ul li.hover {

color: #fff;

background: red;

border: 1px solid red;

border-bottom: 0;

}

ol li {

width: 500px;

background: #ABCDEF;

}

ol li.show {

display: block;

}

ol li.hide {

display: none;

}  
 </style>

<div id="tab">

<ul>

<li>tab1</li>

<li>tab2</li>

<li>tab3</li>

</ul>

<ol>

<li><img src="img/1.jpg"/></li>

<li><img src="img/5.jpg"/></li>

<li><img src="img/3.jpg"/></li>

</ol>

</div>

<script type="text/javascript">

var $uli = $("#tab ul li");

var $oli = $("#tab ol li");

$oli.addClass('hide'); $oli.eq(0).removeClass().addClass('show');

$uli.addClass('normal');  
$uli.eq(0).removeClass().addClass('hover')

$uli.mouseover(function(){

var $index = $(this).index();  
$(this).removeClass().addClass('hover').siblings().removeClass().addClass('normal');  
$oli.eq($index).removeClass().addClass('show').siblings().removeClass().addClass('hide')

});

</script>

1. 动画
   1. 显隐动画
      1. show():显示  
         hide():隐藏  
         原理：hide()通过改变元素的高度宽度和不透明度，直到这三个属性值到0；show()从上到下增加元素的高度，从左到右增加元素宽度，从0到1增加透明度，直至内容完全可见。  
         参数：  
         show()  
         show(speed,callback)  
         speed:字符串或数字，表示动画将运行多久（slow=0.6/normal=0.4/fast=0.2）  
         callback:动画完成时执行的方法
   2. 显隐切换
      1. toggle():切换元素的可见状态  
         原理：匹配元素的宽度、高度以及不透明度，同时进行动画，隐藏动画后将display设置为none  
         参数：  
          toggle(speed)  
          toggle(speed,callback)  
          toggle(boolean)  
         speed:字符串或数字，表示动画将运行多久（slow=0.6/normal=0.4/fast=0.2）  
         easing:使用哪个缓冲函数来过渡的字符串（linear/swing）  
         callback:动画完成时执行的方法  
         boolean:true为显示 false为隐藏
   3. 滑动
      1. 显隐滑动效果  
         slideDown():滑动隐藏  
         slideUp():滑动显示  
         参数：  
         slideDown(speed,callback)  
         slideUp(speed,callback)
      2. 显隐切换滑动  
         slideToggle():显隐滑动切换  
         参数：  
         slideUp(speed,callback)
   4. 渐变：通过改变不透明度
      1. 淡入淡出  
         fadeIn()  
         fadeOut()  
         参数：  
         fadeIn(speed,callback)  
         fadeOut(speed,callback)
      2. 设置淡出透明效果  
         fadeTo():以渐进的方式调整到指定透明度  
         参数：  
         fadeTo(speed,opacity,callback)
      3. 渐变切换：结合fadeIn和fadeOut  
         fadeToggle()  
         参数：  
         fadeOut(speed,callback)
   5. 自定义动画：animate()  
      用animate模拟show():  
       show:表示由透明到不透明  
       toggle:切换  
       hide:表示由显示到隐藏

第四节：应用

1. 树形目录

第五节：综合应用

1. 微信飞机大战案例改造