jQuery性能优化和技巧

# jQuery性能优化

第一条：使用最新的jQuery库类

第二条：选择合适的选择器。$("#id")选择器性能最好；$("p")标签选择器性能也不错。这些选择器在js中有直接实现，而对于属性定位等选择器，js没有具体实现，jQuery要通过搜索每一个元素来定位元素，性能不较低。建议：尽量使用ID选择器，尽量给选择器定义上下文。

第三条：缓存对象,把多次使用的对象，先创建出来。var $obj=$("#id");

第四条：尽量减少循环DOM操作.(多次处理字符串，一次DOM.)。例如，在循环中追加元素。可以再循环中拼接，然后一次DOM追加

第五条：尽量使用原生的JavaScript.

|  |
| --- |
| #for替换each  for(var i=0;i<array.length;i++){ ......}  $.each(array,function(i){  ...  });  $.each(["Aaron", "慕课网"], function(index, value) {  //index是索引,也就是数组的索引  //value就是数组中的值了  });  #cr.checked 替换is（"checked") |

第六条：使用事件代理，$('#myTable td').click(function(){ ......});//有多少个td就会绑定多少个事件。可以使用

$('#myTable td').on('click','td',function(){ ......});

第七条：使用join来拼接字符串。

第八条：压缩Javascript

# jQuery技巧

|  |
| --- |
| #禁止网页右键菜单  $(document).ready(function(){  $(document).on("contextmenu",function(e){  return false;  });  });  #输入框文字获取和失去焦点  $(document).ready(function() {  $("input.text1").val("Enter your search text here.");  textFill( $('input.text1') );  });  function textFill(input){ //input focus text function  var originalvalue = input.val();  input.focus( function(){  if( $.trim(input.val()) == originalvalue ){  input.val('');  }  }).blur( function(){  if( $.trim(input.val()) == '' ){  input.val(originalvalue);  }  });  #获取鼠标位置  $(document).ready(function() {  $(document).mousemove(function(e) {  $('#XY').html("X : " + e.pageX + " | Y : " + e.pageY);  });  });  #判断元素是否存在  $(document).ready(function() {  if ($('#XY').length) {  alert('元素存在!')  } else {  alert('元素不存在!')  }  });  #点击div跳转url  $(document).ready(function() {  $("div").click(function() {  window.location = $(this).find("a").attr("href");  return false;  });  });  #关闭所有的动画  $(document).ready(function() {  jQuery.fx.off=true;  });  #检测鼠标的左键还是右键  $(document).ready(function() {  $("#XY").mousedown(function(e) {  alert(e.which) // 1 = 鼠标左键 ; 2 = 鼠标中键; 3 = 鼠标右键  })  });  #回车提交表单  $(document).ready(function() {  $("input").keyup(function(e) {  if (e.keyCode == "13") {  alert("回车提交!")  }  })  });  #设置全局Ajax参数  $("#load").ajaxStart(function() {  showLoading(); //显示loading  disableButtons(); //禁用按钮  });  $("#load").ajaxComplete(function() {  hideLoading(); //隐藏loading  enableButtons(); //启用按钮  });  function showLoading() {  $("#load").show();  }  function hideLoading() {  $("#load").hide();  }  function disableButtons() {  $("#send1").attr("disabled", "disabled");  }  function enableButtons() {  $("#send1").removeAttr("disabled");  #获取选中的下拉框  function getObj() {  var $obj = $('#someElement').find('option:selected');  alert($obj.val());  }  #切换复选框  var tog = false;  $('button').click(function() {  $("input[type=checkbox]").attr("checked", !tog);  tog = !tog;  });  #扩展String对象的方法  $.extend(String.prototype, {  isPositiveInteger: function() {  return (new RegExp(/^[1-9]\d\*$/).test(this));  },  isInteger: function() {  return (new RegExp(/^\d+$/).test(this));  },  isNumber: function(value, element) {  return (new RegExp(/^-?(?:\d+|\d{1,3}(?:,\d{3})+)(?:\.\d+)?$/).test(this));  },  trim: function() {  return this.replace(/(^\s\*)|(\s\*$)|\r|\n/g, "");  },  trans: function() {  return this.replace(/&lt;/g, '<').replace(/&gt;/g, '>').replace(/&quot;/g, '"');  },  replaceAll: function(os, ns) {  return this.replace(new RegExp(os, "gm"), ns);  },  skipChar: function(ch) {  if (!this || this.length === 0) {  return '';  }  if (this.charAt(0) === ch) {  return this.substring(1).skipChar(ch);  }  return this;  },  isValidPwd: function() {  return (new RegExp(/^([\_]|[a-zA-Z0-9]){6,32}$/).test(this));  },  isValidMail: function() {  return (new RegExp(/^\w+((-\w+)|(\.\w+))\*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)\*\.[A-Za-z0-9]+$/).test(this.trim()));  },  isSpaces: function() {  for (var i = 0; i < this.length; i += 1) {  var ch = this.charAt(i);  if (ch != ' ' && ch != "\n" && ch != "\t" && ch != "\r") {  return false;  }  }  return true;  },  isPhone: function() {  return (new RegExp(/(^([0-9]{3,4}[-])?\d{3,8}(-\d{1,6})?$)|(^\([0-9]{3,4}\)\d{3,8}(\(\d{1,6}\))?$)|(^\d{3,8}$)/).test(this));  },  isUrl: function() {  return (new RegExp(/^[a-zA-z]+:\/\/([a-zA-Z0-9\-\.]+)([-\w .\/?%&=:]\*)$/).test(this));  },  isExternalUrl: function() {  return this.isUrl() && this.indexOf("://" + document.domain) == -1;  }  });  $("button").click(function() {  alert($("input").val().isInteger());  }); |

# AJAX防重复提交的办法总结

Ajax的原理简单来说通过XmlHttpRequest对象来向服务器发异步请求，从服务器获得数据，然后用javascript来操作DOM而更新页面。这其中最关键的一步就是从服务器获得请求数据。

造成重复提交原因：由于AJAX提交数据为异步提交，所以当我们点击提交按钮是通过xmlhttprequest向服务器发送异步请求，发送请求需要有处理时间，我们第一次点击的请求尚未完成，就有接二连三的又提交了几次，同时后来发送的请求同事也被发送到后台处理了，这种情况如果是数据读取则不会有太大影响，但是涉及到数据提交保存或者提交之后多表数据处理就更麻烦了，所以此种情况务必要避免，免得给大家造成麻烦。

## 重复提交解决办法

如果提交对象为按钮的话，可以对按钮设置disabled，此办法适应于按钮提交，此种方法简单粗暴，也是很多人用的办法，代如下：

|  |
| --- |
| //在按钮提交之后和AJAX提交之前将按钮设置为禁用  $("input[type=submit]").attr('disabled',true)  $.ajax({  url:'/post'  data:{a:1,b,1}  success:function(){  //在提交成功之后重新启用该按钮  $("input[type=submit]").attr('disabled',false)  },  error: function(){  //即使AJAX失败也需要将按钮设置为可用状态，因为有可能是网络问题导致的失败，所以需要将按钮设置为可用  $("input[type=submit]").attr('disabled',false)  }  }) |

## 非按钮对象提交

有时候我们在提交数据的时候没有from，只有单个表单组件（input、textarea、radio、checkbox等），所以我们就可以将任何一个对象设置提交对象来完成提交，由于普通对象没有disabled属性，所以我们需要用其它方式来防止重复提交。

|  |
| --- |
| //设置一个对象来控制是否进入AJAX过程  var post\_flag = false;  function post(){  //如果正在提交则直接返回，停止执行  if(post\_flag) return;  //标记当前状态为正在提交状态  post\_flag = true;  $.ajax({//进入AJAX提交过程  url:'/post'  data:{a:1,b,1}  success:function(){  post\_flag =false; //在提交成功之后将标志标记为可提交状态  },  error: function(){  post\_flag =false; //AJAX失败也需要将标志标记为可提交状态  }  })  } |