选择器

- 注意
  - DOM对象，通过原生js获取的是DOM对象，（DOM树上的节点就是DOM对象）。
  - jQuery对象，通过jquery选择器获得的是jQuery对象
  - jQuery对象只能使用jQuery不对象的方法，不能使用DOM对象的方法，同理，DOM对象只能使用DOM的方法不能使用jQuery中对应的方法
- 选择器相互转换
  - DOM对象转换成DOM对象
    - $(DOM对象)
  - jQ对象转换成DOM对象
    - jQuery对象[index]
    - jQuery对象.get(index)
- 基本选择器
  - #id
    - $("#id的ID")
  - .class
    - $(".class的ID")
  - element
    - $("标签名")
  - 群组选择器
    - $("选择器1，选择器2")
      - 注：中间用逗号分隔
  - *
    - $("*")
- 后代选择器
  - $("#id ID")
- 子元素选择器
  - $("#id>ID")
- 紧邻的同辈选择器
  - 忽略空白文本节点
    - $("#first+")
    - $("#first++")
    - $("#first++5")
- 相邻的后面的同辈选择器
  - $("#first~")
    - 后面加标签名，可以获取到相邻的后面的所有此标签名的同辈选择器
- 表单元素选择器
  - 注意，前面一定要加空格
  - :input 获取表单中所有的input,select,textarea元素的注意，前面一定要加空格
    - $("#F1 :input")
  - :text获取所有输入框
    - $("#F1 :text")
  - :password
    - $("#F1 :password")
  - :radio
    - $("#F1 :radio")
  - :checkbox
    - $("#F1 :checkbox")
  - :file
    - $("#F1 :file")
  - :image
    - $("#F1 :image")
  - :hidden
    - $("#F1 :hidden")
  - :submit
    - $("#F1 :submit")
  - :reset
    - $("#F1 :reset")
  - :button
    - $("#F1 :button")

```
<!--表单元素 -->
<input type="text" name="">
<input type="password" name="">
<input type="radio" name="">
<input type="checkbox" name="">
<input type="file" name="">
<input type="image" name="">
<input type="hidden" name="">
<input type="submit" name="">
<input type="reset" name="">
<input type="button" name="">
<select>
    <option>111</option>
    <option>222</option>
</select>
<textarea></textarea>
</form>
```

选择

- 过滤选择器
  - 注意，过滤选择器要和其他选择器结合使用
  - 基本过滤选择器，根据下标来过滤
  - :first 在匹配的所有的集合中选取第一个元素
    - $("li:first")
  - :last 在匹配的所有的集合中选取最后一个元素
    - $("li:last")
  - :eq(index) 下标从0开始
    - $("li:eq(1)")
  - :gt(index) 下标大于index :gt、:great.than
    - $("li:gt(2)")
  - :lt(index) 下标小于index :lt.little.than
    - $("li:lt(1)")
  - :odd 奇数
    - $("li:odd")
  - :even 偶数
    - $("li:even")
  - :eq(数组)选择器数组下标匹配选择器匹配的元素
    - $("li:eq(#)")
  - :header() 获取到标题元素的集合
    - $(":header").css("backgroundColor","red")
  - :animated 匹配正在执行动画的元素
- 内容过滤选择器
  - :contains(指定文本值)
    - 查找所有的包含指定文本值的元素
  - 中转了
    - 查找自定义属性中含有“拼音”字母的，把拼音高亮显示

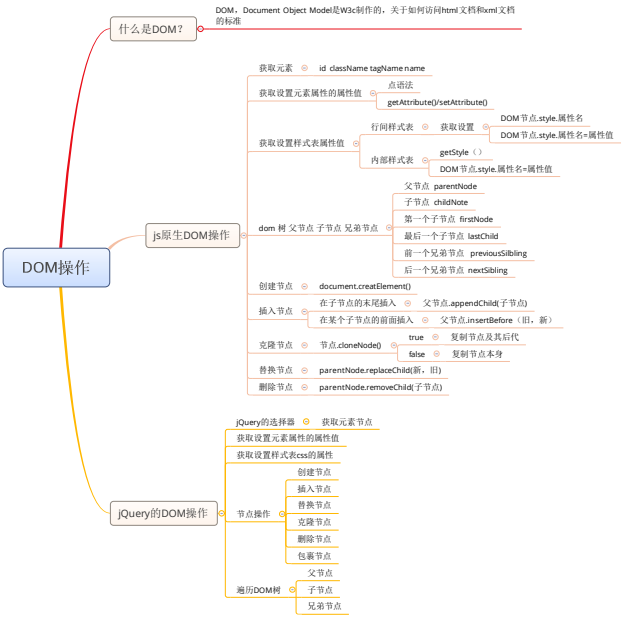
```
var num = $("div:contains(拼音)")
for( var i = 0; i.num.length;i++){
    var div = num[i];
    div.innerHTML div.innerHTML.replace(/拼音/g,"<span style='background:red'>拼音</span>");
}
```

  - :has(selector)
    - 查找所有包含指定选择器的子元素的元素
    - 例子
      - 所有子元素中含有p标签的div
        - $("div:has(p)")
      - 筛选出含有class为test的子元素的div
        - $("div:has(.test)")
  - :empty 匹配空元素
    - 例子
      - 筛选出所有内容为空的div
        - $("div:empty")
  - :parent 筛选出非空元素
    - 例子
      - 筛选出所有内容非空的div
        - $("div:parent")
- 属性过滤选择器
  - [attr]筛选出含有指定属性的元素
    - $("input[placeholder]")
  - [attr=value]筛选出含有指定属性值的元素
    - $("input[type=checkbox]")
  - [attr!=value]筛选出属性值不等于指定值的元素
    - $("input[type!=checkbox]")
  - [attr^=value]筛选出属性值以指定字头的
    - $("[class^=banner]").css("background-color","red");
    - $("[class^=banner]").css("height","50px");
    - $("[class^=banner]").css("width","50px");
  - [attr$=value]筛选出属性值以指定结尾的
    - $("[class$=div]").css("background-color","green");
    - $("[class$=div]").css("height","50px");
    - $("[class$=div]").css("width","50px");
  - [attr*=value]筛选出属性值包含指定值的元素
    - $("[placeholder*=密码]").css("backgroundColor","red");
  - [attr1][attr2]...[attrN]筛选多个指定属性的元素
    - $("input[type][placeholder]").css("background-color","pink");

```
<input type="text" name="" placeholder="请输入内容">
<input type="checkbox" name="">
<input type="checkbox" name="">
<input type="checkbox" name="">
<input type="checkbox" name="">
<input type="text" name="">
<input type="text" name="" placeholder="请输入密码">
<div class="banner_div">ccc</div>
<div class="div">ccc</div>
<div class="div">ccc</div>
```

- 子元素过滤选择器
  - 注意，子元素过滤选择器
  - 需要在前一个selector与子元素过滤选择器之间加上空格
  - :first-child
  - :last-child
  - :nth-child(num)第几元素|:even|:odd 下标是从1开始的
  - :nth-last-child(num)倒数第几元素|:even|:odd 倒数 下标是从1开始的
- 可见性过滤选择器
  - 01 :hidden
    - 匹配所有不可见元素，或者type为hidden的元素
    - $("div:hidden")
  - 02 :visible
    - 匹配所有的可见元素
    - $("div:visible")
- 表单属性过滤选择器
  - 01 :enabled
    - 匹配所有可用元素
    - $(":text:enabled")
  - 02 :disabled
    - 匹配所有不可用的元素
    - $(":text:disabled")
  - 03 :checked
    - 匹配所有选中的表单元素选择，单选框、多选框和下拉option
    - $(":input:checked")
  - 04 :selected
    - 匹配所有选中的option元素
    - $(":option:selected")
  - 案例

```
<form action="">
    <input type="text" name="" disabled="disabled" value="1000">
    <input type="text" name="">
    <input type="text" name="" .checked checkes >
    <input type="checkbox" name="" checked="checked">
    <select id="" multiple="multiple">
        <option value="1" selected="selected">option</option>
        <option value="2">option</option>
        <option value="3" selected="selected">option</option>
    </select>
</form>
<br />
<input type="checkbox" name="" checked="checked">
<br />
<input type="checkbox" name="" checked="checked">
<br />
<input type="checkbox" name="" checked="checked">
<br />
<input type="checkbox" name="" checked="checked">
<br />
<!-- <script type="text/javascript">alert(1);</script>
<script type="text/javascript" src="jquery/jquery-1.8.0.min.js"></script>
<script type="text/javascript">
    console.log(
    console.log("[:text:enabled]",
    console.log("[:text:disabled]",
    console.log("[:checked]",
    console.log("[:option:selected]",
    console.log(
    console.log("[:input:checked:not(form :checked)]",$(
    console.log(
</script>
```


DOM操作
什么是DOM？
DOM，Document Object Model是W3c制作的，关于如何访问html文档和xml文档的标准
js原生DOM操作
获取元素
id
className
tagName
name
获取设置元素属性的属性值
点语法
getAttribute()/setAttribute()
获取设置样式表属性值
行间样式表
获取设置
DOM节点.style.属性名
DOM节点.style.属性名=属性值
内部样式表
getStyle（）
DOM节点.style.属性名=属性值
dom 树 父节点 子节点 兄弟节点
父节点 parentNode
子节点 childNote
第一个子节点 firstNode
最后一个子节点 lastChild
前一个兄弟节点 previousSilbling
后一个兄弟节点 nextSibling
创建节点
document.creatElement()
插入节点
在子节点的末尾插入
父节点.appendChild(子节点)
在某个子节点的前面插入
父节点.insertBefore（旧，新）
克隆节点
节点.cloneNode()
true
复制节点及其后代
false
复制节点本身
替换节点
parentNode.replaceChild(新，旧)
删除节点
parentNode.removeChild(子节点)
jQuery的DOM操作
jQuery的选择器
获取元素节点
获取设置元素属性的属性值
获取设置样式表css的属性
节点操作
创建节点
插入节点
替换节点
克隆节点
删除节点
包裹节点
遍历DOM树
父节点
子节点
兄弟节点

DOM操作

获取设置节点的属性

- 01 $("#.attr()"属性操作")
  - 设置属性的值或者获取属性值。
- 02 $("#.removeAttr()"删属性")
  - 从每一个匹配的元素中删除一个属性
  - $("div").removeAttr("id")
  - 使用匹配的元素集合中的一个属性来设置一个或多个属性。
  - $("#checkbox").attr("checked")
- 03 $("#.prop()"属性")
  - 获取匹配元素集合中的第一个元素的属性值。
  - $("#checkbox").prop("checked",true)
  - 使用匹配元素集合中每个元素的一个或多个属性的一个函数来设置属性值。
  - 实例
    - 代码
      - 复制代码 折叠代码
      - `<input type="checkbox" name="">全选项`
      - `<input type="checkbox" name="">小说项`
      - `<input type="checkbox" name="">小说项`
      - `<input type="checkbox" name="">小说项`
      - `<button onclick="hone()">全选</button>`
      - `<button onclick="no()">取消全选</button>`
      - `<button onclick="reverse()">反选</button>`
      - `<script type="text/javascript">`
      - `function hone(){`
        - `$("checkbox").prop("checked",true)`
      - `}`
      - `function no(){`
        - `$("checkbox").prop("checked",false)`
      - `}`
      - `function reverse(){`
        - `$("checkbox").each(function(){`
          - `this.checked = !this.checked`
        - `})`
      - `}`
      - `</script>`
- 04 $("#.removeProp()"删属性")
  - 删除匹配元素集合的属性的值。

样式/尺寸/定位/位置

- 01 $("#.html()")
  - 向匹配元素集合中设置一个内容。
  - 语法 $("div").html("<h1>hello</h1>")
- 02 $("#.text()")
  - 向匹配元素集合中设置一个文本内容。
  - 语法 $("div").text("123")
- 03 $("#.val()")
  - 向匹配元素集合中设置一个值。
  - 语法 $("textarea").val($("#city").val)

常用的Dom操作的属性

- 01 $("#.offset()")
  - 属性 offset 可以获取，可以设置
  - 语法 $("#parent").offset({left:300,top:300});//offset 可以设置
  - 属性 获取匹配元素集合（分组）的偏移。
  - position 只可以获取 不能设置
- 02 $("#.position()")
  - 语法 $("#son").position();//position 只可以获取 不能设置
- 03 $("#.height()")
  - 获取匹配元素集合的高度（px）。
  - 语法 $("#parent").height(200)
- 04 $("#.width()")
  - 获取匹配元素集合的宽度（px）。
  - 语法 $("#parent").width(200)
  - 实例
    - 代码
      - 复制代码 折叠代码
      - `<meta charset="UTF-8" />`
      - `<meta name="viewport" content="width=device-width, initial-scale=1.0" />`
      - `<meta http-equiv="X-UA-Compatible" content="ie=edge" />`
      - `<title>Document</title>`
      - `<style>`
      - `*{`
        - `margin:0;`
        - `padding:0;`
        - `box-sizing:border-box;`
        - `-webkit-box-sizing:border-box;`
        - `-moz-box-sizing:border-box;`
        - `-ms-box-sizing:border-box;`
        - `-o-box-sizing:border-box;`
      - `}`
      - `#parent{`
        - `width:300px;`
        - `height:300px;`
        - `position:relative;`
        - `margin:100px auto;`
        - `background:red;`
      - `}`
      - `#son{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:blue;`
      - `}`
      - `#son1{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:green;`
      - `}`
      - `#son2{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:yellow;`
      - `}`
      - `#son3{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:pink;`
      - `}`
      - `#son4{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:purple;`
      - `}`
      - `#son5{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:orange;`
      - `}`
      - `#son6{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:black;`
      - `}`
      - `#son7{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:brown;`
      - `}`
      - `#son8{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:gray;`
      - `}`
      - `#son9{`
        - `width:100px;`
        - `height:100px;`
        - `position:absolute;`
        - `left:100px;`
        - `top:100px;`
        - `background:white;`
      - `}`
      - `</style>`

节点操作

- 01 创建元素节点
  - var 变量名 = $(" ")
- 02 插入元素节点
  - 将变量名追加到元素最后入 变量名.appendTo($("#div"))
  - $("div").appendTo($("body"))
  - 将变量名追加到元素最后入 变量名.appendTo($("#div"))
  - $("body").prependTo($("div"))
  - 将变量名追加到元素最前入 变量名.prependTo($("#div"))
  - $("body").prepend($("div"))
  - 将变量名追加到元素最前入 变量名.prependTo($("#div"))
  - $("div").after($a)
  - 将变量名追加到元素后面的兄弟入 变量名.insertAfter($("#div"))
  - $a.insertAfter($("div"))
  - 将变量名追加到元素后面的兄弟入 $(this).before($("#div"))
  - $("div").before($("p1p2p3"))
  - 将变量名追加到元素前面的兄弟入 $(this).insertBefore($("#div"))
  - 将变量名追加到元素前面的兄弟入 $("<p1p2p3>").insertBefore($("div"))
- 03 删除节点
  - $("div").remove()
  - 从DOM中删除所匹配的元素。
  - $("div").empty()
  - 从DOM中删除所有匹配的元素的后代。 删除匹配元素集合的所有子元素
  - $("div").detach()
  - 从DOM中删除所匹配的元素。
  - $("div").empty()
  - 从DOM中删除所有匹配的元素的后代。
  - $("div").remove();
  - 删除匹配元素集合的所有子节点（包括文本和注释）。
  - 实例
    - 代码
      - 复制代码 折叠代码
      - `var $a=new-$("div");`
      - `$a.appendTo($("body"));`
      - `var $b=new-$("div");`
      - `$b.appendTo($("body"));`
      - `var $c=new-$("div");`
      - `$c.appendTo($("body"));`
      - `var $d=new-$("div");`
      - `$d.appendTo($("body"));`
      - `var $e=new-$("div");`
      - `$e.appendTo($("body"));`
      - `var $f=new-$("div");`
      - `$f.appendTo($("body"));`
      - `var $g=new-$("div");`
      - `$g.appendTo($("body"));`
      - `var $h=new-$("div");`
      - `$h.appendTo($("body"));`
      - `var $i=new-$("div");`
      - `$i.appendTo($("body"));`
      - `var $j=new-$("div");`
      - `$j.appendTo($("body"));`
      - `var $k=new-$("div");`
      - `$k.appendTo($("body"));`
      - `var $l=new-$("div");`
      - `$l.appendTo($("body"));`
      - `var $m=new-$("div");`
      - `$m.appendTo($("body"));`
      - `var $n=new-$("div");`
      - `$n.appendTo($("body"));`
      - `var $o=new-$("div");`
      - `$o.appendTo($("body"));`
      - `var $p=new-$("div");`
      - `$p.appendTo($("body"));`
      - `var $q=new-$("div");`
      - `$q.appendTo($("body"));`
      - `var $r=new-$("div");`
      - `$r.appendTo($("body"));`
      - `var $s=new-$("div");`
      - `$s.appendTo($("body"));`
      - `var $t=new-$("div");`
      - `$t.appendTo($("body"));`
      - `var $u=new-$("div");`
      - `$u.appendTo($("body"));`
      - `var $v=new-$("div");`
      - `$v.appendTo($("body"));`
      - `var $w=new-$("div");`
      - `$w.appendTo($("body"));`
      - `var $x=new-$("div");`
      - `$x.appendTo($("body"));`
      - `var $y=new-$("div");`
      - `$y.appendTo($("body"));`
      - `var $z=new-$("div");`
      - `$z.appendTo($("body"));`
      - `var $aa=new-$("div");`
      - `$aa.appendTo($("body"));`
      - `var $ab=new-$("div");`
      - `$ab.appendTo($("body"));`
      - `var $ac=new-$("div");`
      - `$ac.appendTo($("body"));`
      - `var $ad=new-$("div");`
      - `$ad.appendTo($("body"));`
      - `var $ae=new-$("div");`
      - `$ae.appendTo($("body"));`
      - `var $af=new-$("div");`
      - `$af.appendTo($("body"));`
      - `var $ag=new-$("div");`
      - `$ag.appendTo($("body"));`
      - `var $ah=new-$("div");`
      - `$ah.appendTo($("body"));`
      - `var $ai=new-$("div");`
      - `$ai.appendTo($("body"));`
      - `var $aj=new-$("div");`
      - `$aj.appendTo($("body"));`
      - `var $ak=new-$("div");`
      - `$ak.appendTo($("body"));`
      - `var $al=new-$("div");`
      - `$al.appendTo($("body"));`
      - `var $am=new-$("div");`
      - `$am.appendTo($("body"));`
      - `var $an=new-$("div");`
      - `$an.appendTo($("body"));`
      - `var $ao=new-$("div");`
      - `$ao.appendTo($("body"));`
      - `var $ap=new-$("div");`
      - `$ap.appendTo($("body"));`
      - `var $aq=new-$("div");`
      - `$aq.appendTo($("body"));`
      - `var $ar=new-$("div");`
      - `$ar.appendTo($("body"));`
      - `var $as=new-$("div");`
      - `$as.appendTo($("body"));`
      - `var $at=new-$("div");`
      - `$at.appendTo($("body"));`
      - `var $au=new-$("div");`
      - `$au.appendTo($("body"));`
      - `var $av=new-$("div");`
      - `$av.appendTo($("body"));`
      - `var $aw=new-$("div");`
      - `$aw.appendTo($("body"));`
      - `var $ax=new-$("div");`
      - `$ax.appendTo($("body"));`
      - `var $ay=new-$("div");`
      - `$ay.appendTo($("body"));`
      - `var $az=new-$("div");`
      - `$az.appendTo($("body"));`
      - `var $ba=new-$("div");`
      - `$ba.appendTo($("body"));`
      - `var $bb=new-$("div");`
      - `$bb.appendTo($("body"));`
      - `var $bc=new-$("div");`
      - `$bc.appendTo($("body"));`
      - `var $bd=new-$("div");`
      - `$bd.appendTo($("body"));`
      - `var $be=new-$("div");`
      - `$be.appendTo($("body"));`
      - `var $bf=new-$("div");`
      - `$bf.appendTo($("body"));`
      - `var $bg=new-$("div");`
      - `$bg.appendTo($("body"));`
      - `var $bh=new-$("div");`
      - `$bh.appendTo($("body"));`
      - `var $bi=new-$("div");`
      - `$bi.appendTo($("body"));`
      - `var $bj=new-$("div");`
      - `$bj.appendTo($("body"));`
      - `var $bk=new-$("div");`
      - `$bk.appendTo($("body"));`
      - `var $bl=new-$("div");`
      - `$bl.appendTo($("body"));`
      - `var $bm=new-$("div");`
      - `$bm.appendTo($("body"));`
      - `var $bn=new-$("div");`
      - `$bn.appendTo($("body"));`
      - `var $bo=new-$("div");`
      - `$bo.appendTo($("body"));`
      - `var $bp=new-$("div");`
      - `$bp.appendTo($("body"));`
      - `var $bq=new-$("div");`
      - `$bq.appendTo($("body"));`
      - `var $br=new-$("div");`
      - `$br.appendTo($("body"));`
      - `var $bs=new-$("div");`
      - `$bs.appendTo($("body"));`
      - `var $bt=new-$("div");`
      - `$bt.appendTo($("body"));`
      - `var $bu=new-$("div");`
      - `$bu.appendTo($("body"));`
      - `var $bv=new-$("div");`
      - `$bv.appendTo($("body"));`
      - `var $bw=new-$("div");`
      - `$bw.appendTo($("body"));`
      - `var $bx=new-$("div");`
      - `$bx.appendTo($("body"));`
      - `var $by=new-$("div");`
      - `$by.appendTo($("body"));`
      - `var $bz=new-$("div");`
      - `$bz.appendTo($("body"));`
      - `var $ca=new-$("div");`
      - `$ca.appendTo($("body"));`
      - `var $cb=new-$("div");`
      - `$cb.appendTo($("body"));`
      - `var $cc=new-$("div");`
      - `$cc.appendTo($("body"));`
      - `var $cd=new-$("div");`
      - `$cd.appendTo($("body"));`
      - `var $ce=new-$("div");`
      - `$ce.appendTo($("body"));`
      - `var $cf=new-$("div");`
      - `$cf.appendTo($("body"));`
      - `var $cg=new-$("div");`
      - `$cg.appendTo($("body"));`
      - `var $ch=new-$("div");`
      - `$ch.appendTo($("body"));`
      - `var $ci=new-$("div");`
      - `$ci.appendTo($("body"));`
      - `var $cj=new-$("div");`
      - `$cj.appendTo($("body"));`
      - `var $ck=new-$("div");`
      - `$ck.appendTo($("body"));`
      - `var $cl=new-$("div");`
      - `$cl.appendTo($("body"));`
      - `var $cm=new-$("div");`
      - `$cm.appendTo($("body"));`
      - `var $cn=new-$("div");`
      - `$cn.appendTo($("body"));`
      - `var $co=new-$("div");`
      - `$co.appendTo($("body"));`
      - `var $cp=new-$("div");`
      - `$cp.appendTo($("body"));`
      - `var $cq=new-$("div");`
      - `$cq.appendTo($("body"));`
      - `var $cr=new-$("div");`
      - `$cr.appendTo($("body"));`
      - `var $cs=new-$("div");`
      - `$cs.appendTo($("body"));`
      - `var $ct=new-$("div");`
      - `$ct.appendTo($("body"));`
      - `var $cu=new-$("div");`
      - `$cu.appendTo($("body"));`
      - `var $cv=new-$("div");`
      - `$cv.appendTo($("body"));`
      - `var $cw=new-$("div");`
      - `$cw.appendTo($("body"));`
      - `var $cx=new-$("div");`
      - `$cx.appendTo($("body"));`
      - `var $cy=new-$("div");`
      - `$cy.appendTo($("body"));`
      - `var $cz=new-$("div");`
      - `$cz.appendTo($("body"));`
      - `var $da=new-$("div");`
      - `$da.appendTo($("body"));`
      - `var $db=new-$("div");`
      - `$db.appendTo($("body"));`
      - `var $dc=new-$("div");`
      - `$dc.appendTo($("body"));`
      - `var $dd=new-$("div");`
      - `$dd.appendTo($("body"));`
      - `var $de=new-$("div");`
      - `$de.appendTo($("body"));`
      - `var $df=new-$("div");`
      - `$df.appendTo($("body"));`
      - `var $dg=new-$("div");`
      - `$dg.appendTo($("body"));`
      - `var $dh=new-$("div");`
      - `$dh.appendTo($("body"));`
      - `var $di=new-$("div");`
      - `$di.appendTo($("body"));`
      - `var $dj=new-$("div");`
      - `$dj.appendTo($("body"));`
      - `var $dk=new-$("div");`
      - `$dk.appendTo($("body"));`
      - `var $dl=new-$("div");`
      - `$dl.appendTo($("body"));`
      - `var $dm=new-$("div");`
      - `$dm.appendTo($("body"));`
      - `var $dn=new-$("div");`
      - `$dn.appendTo($("body"));`
      - `var $do=new-$("div");`
      - `$do.appendTo($("body"));`
      - `var $dp=new-$("div");`
      - `$dp.appendTo($("body"));`
      - `var $dq=new-$("div");`
      - `$dq.appendTo($("body"));`
      - `var $dr=new-$("div");`
      - `$dr.appendTo($("body"));`
      - `var $ds=new-$("div");`
      - `$ds.appendTo($("body"));`
      - `var $dt=new-$("div");`
      - `$dt.appendTo($("body"));`
      - `var $du=new-$("div");`
      - `$du.appendTo($("body"));`
      - `var $dv=new-$("div");`
      - `$dv.appendTo($("body"));`
      - `var $dw=new-$("div");`
      - `$dw.appendTo($("body"));`
      - `var $dx=new-$("div");`
      - `$dx.appendTo($("body"));`
      - `var $dy=new-$("div");`
      - `$dy.appendTo($("body"));`
      - `var $dz=new-$("div");`
      - `$dz.appendTo($("body"));`

克隆节点

- clone(true/false)
- 克隆节点Dom，同时克隆绑定在节点的事件

```
$("#show").click(function(){
    var $clone=$("div").clone(true);
    $clone.appendTo("body");
});

$("div").click(function(){
    $(this).css("background-color","green");
});
```

替换节点

- 替换匹配的元素集合的内容。
  - 01 将第A.replaceWith(第B元素)
  - 02 将第A.replaceWith(函数)

```
$("#show").click(function(){
    var $a=$("p:eq(0)");
    $a.replaceWith(function(){
        return "<div>replaced</div>";
    });
});
```

包裹节点

- 01 wrap()
  - 将匹配元素集合中的每个元素用其他结构来包裹。
- 02 unwrap()
  - 移除匹配元素的父元素。
- 03 wrapAll()
  - 将匹配元素集合中的全部元素包裹。
  - 将第A.wrapAll(第B)
- 04 wrapInner()
  - 将匹配元素集合中的每个元素的内容包裹。（不包括文本和注释）。
  - 将第A.wrapInner(第B)
  - 实例
    - 代码
      - 复制代码 折叠代码
      - `$("#show").click(function){`
        - `$("div").wrap("<p></p>");`
      - `});`
      - `$("#show").click(function){`
        - `$("div").unwrap();`
      - `});`
      - `$("#show").click(function){`
        - `$("div").wrapAll("<p></p>");`
      - `});`
      - `$("#show").click(function){`
        - `$("div").wrapInner("<p></p>");`
      - `});`

遍历DOM树

- find"的选择器"
  - 查找匹配的每一个元素的后代选择的一个选择器。
  - 实例
    - 代码
      - 复制代码 折叠代码
      - `var $a=$("div").find("p");`
      - `var $b=$("div").find("p:eq(0)");`
      - `var $c=$("div").find("p:eq(1)");`
      - `var $d=$("div").find("p:eq(2)");`
      - `var $e=$("div").find("p:eq(3)");`
      - `var $f=$("div").find("p:eq(4)");`
      - `var $g=$("div").find("p:eq(5)");`
      - `var $h=$("div").find("p:eq(6)");`
      - `var $i=$("div").find("p:eq(7)");`
      - `var $j=$("div").find("p:eq(8)");`
      - `var $k=$("div").find("p:eq(9)");`
      - `var $l=$("div").find("p:eq(10)");`
      - `var $m=$("div").find("p:eq(11)");`
      - `var $n=$("div").find("p:eq(12)");`
      - `var $o=$("div").find("p:eq(13)");`
      - `var $p=$("div").find("p:eq(14)");`
      - `var $q=$("div").find("p:eq(15)");`
      - `var $r=$("div").find("p:eq(16)");`
      - `var $s=$("div").find("p:eq(17)");`
      - `var $t=$("div").find("p:eq(18)");`
      - `var $u=$("div").find("p:eq(19)");`
      - `var $v=$("div").find("p:eq(20)");`
      - `var $w=$("div").find("p:eq(21)");`
      - `var $x=$("div").find("p:eq(22)");`
      - `var $y=$("div").find("p:eq(23)");`
      - `var $z=$("div").find("p:eq(24)");`
      - `var $aa=$("div").find("p:eq(25)");`
      - `var $ab=$("div").find("p:eq(26)");`
      - `var $ac=$("div").find("p:eq(27)");`
      - `var $ad=$("div").find("p:eq(28)");`
      - `var $ae=$("div").find("p:eq(29)");`
      - `var $af=$("div").find("p:eq(30)");`
      - `var $ag=$("div").find("p:eq(31)");`
      - `var $ah=$("div").find("p:eq(32)");`
      - `var $ai=$("div").find("p:eq(33)");`
      - `var $aj=$("div").find("p:eq(34)");`
      - `var $ak=$("div").find("p:eq(35)");`
      - `var $al=$("div").find("p:eq(36)");`
      - `var $am=$("div").find("p:eq(37)");`
      - `var $an=$("div").find("p:eq(38)");`
      - `var $ao=$("div").find("p:eq(39)");`
      - `var $ap=$("div").find("p:eq(40)");`
      - `var $aq=$("div").find("p:eq(41)");`
      - `var $ar=$("div").find("p:eq(42)");`
      - `var $as=$("div").find("p:eq(43)");`
      - `var $at=$("div").find("p:eq(44)");`
      - `var $au=$("div").find("p:eq(45)");`
      - `var $av=$("div").find("p:eq(46)");`
      - `var $aw=$("div").find("p:eq(47)");`
      - `var $ax=$("div").find("p:eq(48)");`
      - `var $ay=$("div").find("p:eq(49)");`
      - `var $az=$("div").find("p:eq(50)");`
      - `var $ba=$("div").find("p:eq(51)");`
      - `var $bb=$("div").find("p:eq(52)");`
      - `var $bc=$("div").find("p:eq(53)");`
      - `var $bd=$("div").find("p:eq(54)");`
      - `var $be=$("div").find("p:eq(55)");`
      - `var $bf=$("div").find("p:eq(56)");`
      - `var $bg=$("div").find("p:eq(57)");`
      - `var $bh=$("div").find("p:eq(58)");`
      - `var $bi=$("div").find("p:eq(59)");`
      - `var $bj=$("div").find("p:eq(60)");`
      - `var $bk=$("div").find("p:eq(61)");`
      - `var $bl=$("div").find("p:eq(62)");`
      - `var $bm=$("div").find("p:eq(63)");`
      - `var $bn=$("div").find("p:eq(64)");`
      - `var $bo=$("div").find("p:eq(65)");`
      - `var $bp=$("div").find("p:eq(66)");`
      - `var $bq=$("div").find("p:eq(67)");`
      - `var $br=$("div").find("p:eq(68)");`
      - `var $bs=$("div").find("p:eq(69)");`
      - `var $bt=$("div").find("p:eq(70)");`
      - `var $bu=$("div").find("p:eq(71)");`
      - `var $bv=$("div").find("p:eq(72)");`
      - `var $bw=$("div").find("p:eq(73)");`
      - `var $bx=$("div").find("p:eq(74)");`
      - `var $by=$("div").find("p:eq(75)");`
      - `var $bz=$("div").find("p:eq(76)");`
      - `var $ca=$("div").find("p:eq(77)");`
      - `var $cb=$("div").find("p:eq(78)");`
      - `var $cc=$("div").find("p:eq(79)");`
      - `var $cd=$("div").find("p:eq(80)");`
      - `var $ce=$("div").find("p:eq(81)");`
      - `var $cf=$("div").find("p:eq(82)");`
      - `var $cg=$("div").find("p:eq(83)");`
      - `var $ch=$("div").find("p:eq(84)");`
      - `var $ci=$("div").find("p:eq(85)");`
      - `var $cj=$("div").find("p:eq(86)");`
      - `var $ck=$("div").find("p:eq(87)");`
      - `var $cl=$("div").find("p:eq(88)");`
      - `var $cm=$("div").find("p:eq(89)");`
      - `var $cn=$("div").find("p:eq(90)");`
      - `var $co=$("div").find("p:eq(91)");`
      - `var $cp=$("div").find("p:eq(92)");`
      - `var $cq=$("div").find("p:eq(93)");`
      - `var $cr=$("div").find("p:eq(94)");`
      - `var $cs=$("div").find("p:eq(95)");`
      - `var $ct=$("div").find("p:eq(96)");`
      - `var $cu=$("div").find("p:eq(97)");`
      - `var $cv=$("div").find("p:eq(98)");`
      - `var $cw=$("div").find("p:eq(99)");`
      - `var $cx=$("div").find("p:eq(100)");`
      - `var $cy=$("div").find("p:eq(101)");`
      - `var $cz=$("div").find("p:eq(102)");`
      - `var $da=$("div").find("p:eq(103)");`
      - `var $db=$("div").find("p:eq(104)");`
      - `var $dc=$("div").find("p:eq(105)");`
      - `var $dd=$("div").find("p:eq(106)");`
      - `var $de=$("div").find("p:eq(107)");`
      - `var $df=$("div").find("p:eq(108)");`
      - `var $dg=$("div").find("p:eq(109)");`
      - `var $dh=$("div").find("p:eq(110)");`
      - `var $di=$("div").find("p:eq(111)");`
      - `var $dj=$("div").find("p:eq(112)");`
      - `var $dk=$("div").find("p:eq(113)");`
      - `var $dl=$("div").find("p:eq(114)");`
      - `var $dm=$("div").find("p:eq(115)");`
      - `var $dn=$("div").find("p:eq(116)");`
      - `var $do=$("div").find("p:eq(117)");`
      - `var $dp=$("div").find("p:eq(118)");`
      - `var $dq=$("div").find("p:eq(119)");`
      - `var $dr=$("div").find("p:eq(120)");`
      - `var $ds=$("div").find("p:eq(121)");`
      - `var $dt=$("div").find("p:eq(122)");`
      - `var $du=$("div").find("p:eq(123)");`
      - `var $dv=$("div").find("p:eq(124)");`
      - `var $dw=$("div").find("p:eq(125)");`
      - `var $dx=$("div").find("p:eq(126)");`
      - `var $dy=$("div").find("p:eq(127)");`
      - `var $dz=$("div").find("p:eq(128)");`
      - `var $ea=$("div").find("p:eq(129)");`
      - `var $eb=$("div").find("p:eq(130)");`
      - `var $ec=$("div").find("p:eq(131)");`
      - `var $ed=$("div").find("p:eq(132)");`
      - `var $ee=$("div").find("p:eq(133)");`
      - `var $ef=$("div").find("p:eq(134)");`
      - `var $eg=$("div").find("p:eq(135)");`
      - `var $eh=$("div").find("p:eq(136)");`
      - `var $ei=$("div").find("p:eq(137)");`
      - `var $ej=$("div").find("p:eq(138)");`
      - `var $ek=$("div").find("p:eq(139)");`
      - `var $el=$("div").find("p:eq(140)");`
      - `var $em=$("div").find("p:eq(141)");`
      - `var $en=$("div").find("p:eq(142)");`
      - `var $eo=$("div").find("p:eq(143)");`
      - `var $ep=$("div").find("p:eq(144)");`
      - `var $eq=$("div").find("p:eq(145)");`
      - `var $er=$("div").find("p:eq(146)");`
      - `var $es=$("div").find("p:eq(147)");`
      - `var $et=$("div").find("p:eq(148)");`
      - `var $eu=$("div").find("p:eq(149)");`
      - `var $ev=$("div").find("p:eq(150)");`
      - `var $ew=$("div").find("p:eq(151)");`
      - `var $ex=$("div").find("p:eq(152)");`
      - `var $ey=$("div").find("p:eq(153)");`
      - `var $ez=$("div").find("p:eq(154)");`
      - `var $fa=$("div").find("p:eq(155)");`
      - `var $fb=$("div").find("p:eq(156)");`
      - `var $fc=$("div").find("p:eq(157)");`
      - `var $fd=$("div").find("p:eq(158)");`
      - `var $fe=$("div").find("p:eq(159)");`
      - `var $ff=$("div").find("p:eq(160)");`
      - `var $fg=$("div").find("p:eq(161)");`
      - `var $fh=$("div").find("p:eq(162)");`
      - `var $fi=$("div").find("p:eq(163)");`
      - `var $fj=$("div").find("p:eq(164)");`
      - `var $fk=$("div").find("p:eq(165)");`
      - `var $fl=$("div").find("p:eq(166)");`
      - `var $fm=$("div").find("p:eq(167)");`
      - `var $fn=$("div").find("p:eq(168)");`
      - `var $fo=$("div").find("p:eq(169)");`
      - `var $fp=$("div").find("p:eq(170)");`
      - `var $fq=$("div").find("p:eq(171)");`
      - `var $fr=$("div").find("p:eq(172)");`
      - `var $fs=$("div").find("p:eq(173)");`
      - `var $ft=$("div").find("p:eq(174)");`
      - `var $fu=$("div").find("p:eq(175)");`
      - `var $fv=$("div").find("p:eq(176)");`
      - `var $fw=$("div").find("p:eq(177)");`
      - `var $fx=$("div").find("p:eq(178)");`
      - `var $fy=$("div").find("p:eq(179)");`
      - `var $fz=$("div").find("p:eq(180)");`
      - `var $ga=$("div").find("p:eq(181)");`
      - `var $gb=$("div").find("p:eq(182)");`
      - `var $gc=$("div").find("p:eq(183)");`
      - `var $gd=$("div").find("p:eq(184)");`
      - `var $ge=$("div").find("p:eq(185)");`
      - `var $gf=$("div").find("p:eq(186)");`
      - `var $gg=$("div").find("p:eq(187)");`
      - `var $gh=$("div").find("p:eq(188)");`
      - `var $gi=$("div").find("p:eq(189)");`
      - `var $gj=$("div").find("p:eq(190)");`
      - `var $gk=$("div").find("p:eq(191)");`
      - `var $gl=$("div").find("p:eq(192)");`
      - `var $gm=$("div").find("p:eq(193)");`
      - `var $gn=$("div").find("p:eq(194)");`
      - `var $go=$("div").find("p:eq(195)");`
      - `var $gp=$("div").find("p:eq(196)");`
      - `var $gq=$("div").find("p:eq(197)");`
      - `var $gr=$("div").find("p:eq(198)");`
      - `var $gs=$("div").find("p:eq(199)");`
      - `var $gt=$("div").find("p:eq(200)");`
- 选择器
  - 代码
    - 复制代码 折叠代码
    - `var $a=$("div").find("p");`
    - `var $b=$("div").find("p:eq(0)");`
    - `var $c=$("div").find("p:eq(1)");`
    - `var $d=$("div").find("p:eq(2)");`
    - `var $e=$("div").find("p:eq(3)");`
    - `var $f=$("div").find("p:eq(4)");`
    - `var $g=$("div").find("p:eq(5)");`
    - `var $h=$("div").find("p:eq(6)");`
    - `var $i=$("div").find("p:eq(7)");`
    - `var $j=$("div").find("p:eq(8)");`
    - `var $k=$("div").find("p:eq(9)");`
    - `var $l=$("div").find("p:eq(10)");`
    - `var $m=$("div").find("p:eq(11)");`
    - `var $n=$("div").find("p:eq(12)");`
    - `var $o=$("div").find("p:eq(13)");`
    - `var $p=$("div").find("p:eq(14)");`
    - `var $q=$("div").find("p:eq(15)");`
    - `var $r=$("div").find("p:eq(16)");`
    - `var $s=$("div").find("p:eq(17)");`
    - `var $t=$("div").find("p:eq(18)");`
    - `var $u=$("div").find("p:eq(19)");`
    - `var $v=$("div").find("p:eq(20)");`
    - `var $w=$("div").find("p:eq(21)");`
    - `var $x=$("div").find("p:eq(22)");`
    - `var $y=$("div").find("p:eq(23)");`
    - `var $z=$("div").find("p:eq(24)");`
    - `var $aa=$("div").find("p:eq(25)");`
    - `var $ab=$("div").find("p:eq(26)");`
    - `var $ac=$("div").find("p:eq(27)");`
    - `var $ad=$("div").find("p:eq(28)");`
    - `var $ae=$("div").find("p:eq(29)");`
    - `var $af=$("div").find("p:eq(30)");`
    - `var $ag=$("div").find("p:eq(31)");`
    - `var $ah=$("div").find("p:eq(32)");`
    - `var $ai=$("div").find("p:eq(33)");`
    - `var $aj=$("div").find("p:eq(34)");`
    - `var $ak=$("div").find("p:eq(35)");`
    - `var $al=$("div").find("p:eq(36)");`
    - `var $am=$("div").find("p:eq(37)");`
    - `var $an=$("div").find("p:eq(38)");`
    - `var $ao=$("div").find("p:eq(39)");`
    - `var $ap=$("div").find("p:eq(40)");`
    - `var $aq=$("div").find("p:eq(41)");`
    - `var $ar=$("div").find("p:eq(42)");`
    - `var $as=$("div").find("p:eq(43)");`
    - `var $at=$("div").find("p:eq(44)");`
    - `var $au=$("div").find("p:eq(45)");`
    - `var $av=$("div").find("p:eq(46)");`
    - `var $aw=$("div").find("p:eq(47)");`
    - `var $ax=$("div").find("p:eq(48)");`
    - `var $ay=$("div").find("p:eq(49)");`
    - `var $az=$("div").find("p:eq(50)");`
    - `var $ba=$("div").find("p:eq(51)");`
    - `var $bb=$("div").find("p:eq(52)");`
    - `var $bc=$("div").find("p:eq(53)");`
    - `var $bd=$("div").find("p:eq(54)");`
    - `var $be=$("div").find("p:eq(55)");`
    - `var $bf=$("div").find("p:eq(56)");`
    - `var $bg=$("div").find("p:eq(57)");`
    - `var $bh=$("div").find("p:eq(58)");`
    - `var $bi=$("div").find("p:eq(59)");`
    - `var $bj=$("div").find("p:eq(60)");`
    - `var $bk=$("div").find("p:eq(61)");`
    - `var $bl=$("div").find("p:eq(62)");`
    - `var $bm=$("div").find("p:eq(63)");`
    - `var $bn=$("div").find("p:eq(64)");`
    - `var $bo=$("div").find("p:eq(65)");`
    - `var $bp=$("div").find("p:eq(66)");`
    - `var $bq=$("div").find("p:eq(67)");`
    - `var $br=$("div").find("p:eq(68)");`
    - `var $bs=$("div").find("p:eq(69)");`
    - `var $bt=$("div").find("p:eq(70)");`
    - `var $bu=$("div").find("p:eq(71)");`
    - `var $bv=$("div").find("p:eq(72)");`
    - `var $bw=$("div").find("p:eq(73)");`
    - `var $bx=$("div").find("p:eq(74)");`
    - `var $by=$("div").find("p:eq(75)");`
    - `var $bz=$("div").find("p:eq(76)");`
    - `var $ca=$("div").find("p:eq(77)");`
    - `var $cb=$("div").find("p:eq(78)");`
    - `var $cc=$("div").find("p:eq(79)");`
    - `var $cd=$("div").find("p:eq(80)");`
    - `var $ce=$("div").find("p:eq(81)");`
    - `var $cf=$("div").find("p:eq(82)");`
    - `var $cg=$("div").find("p:eq(83)");`
    - `var $ch=$("div").find("p:eq(84)");`
    - `var $ci=$("div").find("p:eq(85)");`
    - `var $cj=$("div").find("p:eq(86)");`
    - `var $ck=$("div").find("p:eq(87)");`
    - `var $cl=$("div").find("p:eq(88)");`
    - `var $cm=$("div").find("p:eq(89)");`
    - `var $cn=$("div").find("p:eq(90)");`
    - `var $co=$("div").find("p:eq(91)");`
    - `var $cp=$("div").find("p:eq(92)");`
    - `var $cq=$("div").find("p:eq(93)");`
    - `var $cr=$("div").find("p:eq(94)");`
    - `var $cs=$("div").find("p:eq(95)");`
    - `var $ct=$("div").find("p:eq(96)");`
    - `var $cu=$("div").find("p:eq(97)");`
    - `var $cv=$("div").find("p:eq(98)");`
    - `var $cw=$("div").find("p:eq(99)");`
    - `var $cx=$("div").find("p:eq(100)");`
    - `var $cy=$("div").find("p:eq(101)");`
    - `var $cz=$("div").find("p:eq(102)");`
    - `var $da=$("div").find("p:eq(103)");`
    - `var $db=$("div").find("p:eq(104)");`
    - `var $dc=$("div").find("p:eq(105)");`
    - `var $dd=$("div").find("p:eq(106)");`
    - `var $de=$("div").find("p:eq(107)");`
    - `var $df=$("div").find("p:eq(108)");`
    - `var $dg=$("div").find("p:eq(109)");`
    - `var $dh=$("div").find("p:eq(110)");`
    - `var $di=$("div").find("p:eq(111)");`
    - `var $dj=$("div").find("p:eq(112)");`
    - `var $dk=$("div").find("p:eq(113)");`
    - `var $dl=$("div").find("p:eq(114)");`
    - `var $dm=$("div").find("p:eq(115)");`
    - `var $dn=$("div").find("p:eq(116)");`
    - `var $do=$("div").find("p:eq(117)");`
    - `var $dp=$("div").find("p:eq(118)");`
    - `var $dq=$("div").find("p:eq(119)");`
    - `var $dr=$("div").find("p:eq(120)");`
    - `var $ds=$("div").find("p:eq(121)");`
    - `var $dt=$("div").find("p:eq(122)");`
    - `var $du=$("div").find("p:eq(123)");`
    - `var $dv=$("div").find("p:eq(124)");`
    - `var $dw=$("div").find("p:eq(125)");`
    - `var $dx=$("div").find("p:eq(126)");`
    - `var $dy=$("div").find("p:eq(127)");`
    - `var $dz=$("div").find("p:eq(128)");`
    - `var $ea=$("div").find("p:eq(129)");`
    - `var $eb=$("div").find("p:eq(130)");`
    - `var $ec=$("div").find("p:eq(131)");`
    - `var $ed=$("div").find("p:eq(132)");`
    - `var $ee=$("div").find("p:eq(133)");`
    - `var $ef=$("div").find("p:eq(134)");`
    - `var $eg=$("div").find("p:eq(135)");`
    - `var $eh=$("div").find("p:eq(136)");`
    - `var $ei=$("div").find("p:eq(137)");`
    - `var $ej=$("div").find("p:eq(138)");`
    - `var $ek=$("div").find("p:eq(139)");`
    - `var $el=$("div").find("p:eq(140)");`
    - `var $em=$("div").find("p:eq(141)");`
    - `var $en=$("div").find("p:eq(142)");`
    - `var

事件

- 原生
  - 事件处理程序
  - 添加事件响应函数的方式
    - HTML事件处理程序
    - DOM 0级事件处理程序
    - DOM 2事件处理程序
  - 事件流
    - 事件冒泡
    - 事件捕获
  - 事件对象
    - 常用的属性和方法
    - 阻止冒泡
    - 阻止默认行为
- jQuery事件
  - 事件基础
    - 添加事件
    - 移除事件
    - 事件简写
    - 复合事件
  - 事件对象
    - 获取事件对象
    - 事件对象常用的属性与方法
      - stopPropagation()
      - preventDafault()
      - target
      - which
      - type 获取事件名
    - 阻止冒泡 阻止默认
  - 事件高级
    - 触发事件
    - 自定义事件
    - 事件委托

动画

- jQuery常用的动画
  - hide（）
  - show（）
  - fadeIn（）
  - fadeOut（）
  - slideDown（）
  - slideUp（）
- 自定义动画
  - animate("最终效果"，时间，回调函数)
    - 回调函数可有可无
    - 简单动画
    - 累加动画
    - 多个动画，顺序执行
    - 综合动画
  - stop（"是否清空动画队列"，"是否显示最终效果"）
  - delay（时间） 推迟执行动画的效果

ajax

- 原生js
- jQuery中的ajax
  - load（）
  - $.get()
  - $.post()
  - $.getScript() 跨域请求
  - $.getJson()
  - $.Ajax()
    - 常用参数
    - 同源请求
    - 子主题 3

DOM

添加事件

- 01 — on("事件名",响应函数)

```
//01 on("事件名",响应函数)
$("div").on("click",function(){
    console.log("放欲施铁哈！")
})
```

- 02 — 给多个事件添加同一个响应函数，事件名与事件名之间用空格隔开 — on("事件名 事件名",响应函数)

```
//02 on("事件名 事件名",响应函数)
//说明：事件名 事件名之间用空格隔开
$("div").on("mouseover click",function(){
    console.log("一毛！")
})
```

- 03 — 一次性添加多个事件响应 — on({"事件名1":"响应函数1","事件名2":"响应函数2","事件名3":"响应函数3"})

```
//03 一次性添加多个事件响应
//说明：{"事件名1":"响应函数1","事件名2":"响应函数2","事件名3":"响应函数3"}
$("div").on({
    "click":function(){console.log("click")},
    "mouseover":function(){console.log("mouseover")},
    "mousout":function(){console.log("mousout")}
})
```

- 04 — 可以向响应函数传参 — 参数写在函数名的后面，用逗号分隔；参数有对象几个参数

```
//04 可以向响应函数传参
$("div").on("click",{"小明":"哈哈"},function(event){
    //事件对象 event.data
    console.log(event.data);
});
```

- 05 — 链式操作 — 参数写在函数名的后面，用逗号分隔

```
//链式操作
$("div").on("click",function(){console.log("click");})
    .on("mouseover",function(){console.log("mouseover");})
    .on("click",hand)
function hand(){console.log("click222");}
```

移除事件

- 01 — 移除所有事件 — off()
- 02 — 移除某一事件 — off（事件名）
- 03 — 移除某一响应函数 — off（“事件名”，函数名）

```
//1 移除所有事件
$("div").off();

//2 移除给事件 off(事件名)
$("div").off("click")

//3 移除某一个响应函数
$("div").off("click",hand)
```

绑定事件

- 绑定事件 — bind（“事件名”，响应函数）
- 移除绑定的所有事件 — unbind（）
- 移除绑定事件 — unbind（“事件名”）

页面载入

- $(document).ready(function(){});
- 语法 — $(function(){ 表示代码应此处添加 }) — 写在加载文件全局
- load事件与ready事件区别 — load是指页面中所有元素加载完毕之后就会触发(所有的文件图片都予载完毕) ; ready是指DOM树已经加载完毕之后就会触发(此时图片不一定加载完)

合成事件

- mouseover和mousout的合成
- hover（）— $("").hover(mouseover的事件函数,mousout的事件函数)

```
$(function(){
//hover 是mouseover和mousout的合成
//$("").hover(mouseover的事件函数,mousout的事件函数)
    $("div").hover(function(){
        $(this).css("background-color","red")
    },function(){
        $(this).css("background-color","green")
    })
})
```

常见事件

- focus — 聚焦
- focusout — 点击
- 案例

```
<!DOCTYPE html>
<html>
    <meta charset="utf-8" />
    <title>事件应用案例</title>
    <meta name="keywords" />
    <meta name="description" />
    <meta name="author" />
    <meta name="viewport" />
    <link rel="stylesheet" />
    <style type="text/css">
    /*
    事件应用应用范围
    应用范围：在input应用聚焦应用最广泛
    1.在登录页面，要求“下拉菜单或
    2.在搜索页面
    3.在登录页面，
    4.input文本框中，鼠标聚入要显示出 会提示“输入工资”（绿色字体）
    5.当你“输入信息”（红色字体）
    */
    .div{
    .input{ type="text" value="输入工资" /> <span id="div" />
    .input{ type="text" value="输入信息" />
    .input{ type="text" />
    .input{ type="text" />
    .input{ type="text
```

中心主题

触发事件

- trigger("事件名")
  - 直接一个匹配的元素上触发复杂事件。

```
$("button").trigger("click")
// 当 button 点击时
$("input").trigger("focus")
```

自定义事件

- 01 记得提前on bind绑定
- 02 通过trigger（）触发函数
- 03

```
$("div").on("slideup",function(){console.log("上滑");});
    .on("slidedown",function(){console.log("下滑");});
    .on("slideleft",function(){console.log("左滑");});
    .on("slideright",function(){console.log("右滑");});

var start0;
var start1;
var end0;
var end1;
//touch事件
$("div").on("mousedown",function(event){
    start0=event.pageX;
    start1=event.pageY;
});
$("div").on("mouseup",function(event){
    end0=event.pageX;
    end1=event.pageY;
    if(end0-start0 >0 && Math.abs(end0-start0)>50){
        $(this).trigger("slideup")
    }
    if(end0-start0 <0 && Math.abs(end0-start0)>50){
        $(this).trigger("slidedown")
    }
    if(end1-start1 >0 && Math.abs(end1-start1)>50){
        $(this).trigger("slideleft")
    }
    if(end1-start1 <0 && Math.abs(end1-start1)>50){
        $(this).trigger("slideright")
    }
});
```

事件委托

- 是对事件处理，利用事件冒泡给父元素添加事件处理程序从而使用等于元素都可以处理该事件
- 优点
  - 1 减少DOM操作，提高交互效率
  - 2 动态加载子元素时可以响应事件
  - 3 连续操作 — 如果给目标子元素绑定处理程序的话要进行计时器可以考虑添加时间交互事，以元素内容子元素动态加载

```
//优化一下，只对目标元素，可以使用冒泡机制来事件
$("table").click(function(){
    //$(this).toggle();//这样会作用于table
    $(e.target).ref("body,tr,table").ess("background-color","red");
});
```

jQuery自带事件

- 01
  - show（） — 设元素显示，相当于 ess("display","block")；
  - hide(slow|normal|num) — 设元素隐藏，相当于 ess("display","none")；

```
$("button:eq(0)").click(function(){
    //hide(slow|normal|num) 元素隐藏 ess("display","none");
    if($("div").ess("display")=="none"){
        $("div").show(1000);
    }else{
        $("div").hide(1000);
    }
})
```

  - 可传时间参数 slow|normal|num
- 02
  - fadeIn( )
  - fadeOut( )
  - fadeTogle(e.) — 通过不透明度的变化来开展淡行百配元素的这入和淡出效果，并且动画完成后可选地触发一个回调函数。

```
//fadeIn(num,callback) 淡入效果 ess(opacity:1)
$("button:eq(1)").click(function(){
    if($("div").ess("opacity")==0.0 ){
        $("div").fadeIn(1000);
    }else{
        $("div").fadeOut(1000);
    }
})
```

  - 可传时间参数 slow|normal|num
  - 通过不透明度的变化来开展淡行百配元素的淡入和淡出效果，并且动画完成后可选地触发一个回调函数。
- 03
  - slideDown()
  - slideUp()
  - slideTogle()
  - 这函数是改变高度的操作百匹配元素的可见性，并且加载完成后可选地触发一个回调函数。

自定义动画

- $("div").animate(属性设置，执行时间，回调函数)；
- 01 顺序执行

```
$("div").animate({"left":"500px"},2000,function(){ console.log("over"); });
$("div").animate({"top":"300px"},2000,function(){console.log("done")});
```

- 02 同时执行

```
//同时执行
$("div").animate({"left":"500px","top":"500px"},2000);
```

- 03 累加动画

```
// 累加动画
// delay(时间)
$("div").delay(2000).animate({"left":"+=500"},2000);
```

- 04 延迟动画

```
delay(时间)，  多长时间之后执行动画。
// 累加动画
// delay(时间)
$("div").delay(2000).animate({"left":"+=500"},2000);
```

  - 注意可以将参数数的代表意义，true表示清空所有队列

- 05 停止动画

```
//stop(是否清空动画队列，是否显示最终效果)
$("div").hover(function(){
    $(this).stop(true);
    $(this).animate({"width":"400px"},1000)
           .animate({"height":"400px"},1000)
},function(){
    $(this).stop(true);//停止队列
    $(this).animate({"width":"200px"},1000)
           .animate({"height":"200px"},1000)
})
```

index（）事件

- 获取当前元素在兄弟集合的下标

```
$("div").click(function(){
    //获取当前子元素在集合中第几个元素，从0开始
    console.log( $(this).index() );
})
```

Ajax

load()

- 指定要加载一段内容
- $obj.load(url)
  - url：一般情况是要加载一个html文件的路径
  - 回调：加载完成后，load()方法会执行一次回调函数（approved?）, $obj
- $obj.load(url, 回调函数)
  - url：指定要加载的一个html文件的路径
  - 回调函数：可以不加参数或加参数，可以在函数内部对文档进行一些处理

```
$("button").click(function(){
  $("#cont").load("xd.html",function(){
    $("li").eq(1).eq("background-color","red")
  })
})
```

点击按钮，会将名为id为#cont的div加载进来，并且将div内的第二个li的背景颜色变成红色

- 同步

- 异步

$.get()

- 本身函数
- $.get(URL, data, fn, dataType)
  - URL：请求文件的地址
  - 本身
  - data：请求参数 {}
  - fn：回调函数
  - dataType：文件类型
    - text | xml | json | jsonp | script | html
- 本身
- 别名
- 回调
  - success：成功
  - error：失败
  - 请求
  - textStatus的三种状态
    - not found：找不到文件
    - notfind find：找到文件

```
$.get("tan.txt",function(data,textStatus,fn){
//success  ajaxSuccess,ajaxComplete
//textStatus: timeout
//error    ajaxError,ajaxComplete
//  参数    说明    可选值和值
//textStatus  "success"|
//  error,ajaxComplete;
},
"text"
)
```

- 别名
- 同步

```
{"button":$.click(function(){
    $.get("get_html.php",
    {
      "username":$("#username").val(),
      "content":$("#cont").val()
    },
    function(data,textStatus,xh){
      if(textStatus=="success"){
        $("#msg").html(data.content);
        if($("#p".data.username=="yjh"){$.appendTo($("#msg"));
        if($("#p".data.content=="yjh"){$.appendTo($("#msg"));
      }
      else{
      }
    }
    )
})
}
"html"
)
```

点击按钮，将数据发送至一个jsmp文件，并将请求成功写入名为#msg的div内

- 对象 ☆

$.post()

- $.post(url,data,fn,dataType)
  - 别名
  - 本身
  - 别名
  - 别名
  - 别名

ES6新特性

变量

- 新增 let 只在块级变量范围内使用的变量
  - 用于声明块级作用域变量的，只在大括号里可以使用
  - 1.在同一作用域内，不允许声明两遍相同
  - 2.不存在变量提升

```
var a = 10;
var b = 20;
console.log(a);//10
console.log(b);//20
console.log(c);//undefined
var c = 30;
console.log(c);//30
```

- 区别 var
  - var

```
var i = 5;
var str = '';
for (var i = 0; i < 3; i++) {
  console.log('i: ' + i);
  var i = 'abc';
  str += i + ';';
}
console.log('str: ' + str);
// i: 0
// i: 1
// i: 2
// str: abc;abc;abc;
```

  - let

```
let i = 5;
let str = '';
for (let i = 0; i < 3; i++) {
  console.log('i: ' + i);
  let i = 'abc';
  str += i + ';';
}
console.log('str: ' + str);
// i: 0
// i: 1
// i: 2
// str: 0;1;2;
```

- 新增 const 新增块级作用域变量
  - 定义只读变量
  - const
    - 1.const声明变量必须立即赋值
    - 2.const声明的变量在一个块级作用域有效
    - 3.const声明的变量在"变量提前声明"，会导致声明区效用
    - 4.const声明变量不能（修改）但 var var-const声明的变量是可

```
const SERVER = '1.2.3'
console.log(SERVER);//'1.2.3'
SERVER = '1.2.4';//报错 Uncaught TypeError: Assignment to constant variable.
```

解构赋值

- 数组解构赋值 =>

```
var [a,b,c,d] = [1,2,3,4];
console.log(a,b,c,d);

var [a,b,c,d,e] = [1,2,3,4];
console.log(a,b,c,d,e);//undefined
```

- 对象解构赋值 =>

```
var {name,age} = {name:"张三",age:"18"}
console.log(name,age)
```

- 嵌套解构赋值 =>

```
var [a, [b, c], d] = [1, [2, 3], 4];
console.log(a, b, c, d)
// 1 2 3 4

var [a, [b, [c, d], e], f, g] = [1, [2, [3, 4], 5], 6, 7]
console.log(a, b, c, d, e, f, g)
// 1 2 3 4 5 6 7

var [a, [b, [c, d], e], f, g] = [1, [2, [3, 4], 5], 6, 7]
console.log(a, b, c, d, e, f, g)
// 1 2 3 4 5 6 7
```

- 解构赋值在函数中的应用 =>

```
function add([x, y]){
  return x + y;
}
console.log(add([1, 2]));//3
console.log(add([3, 7]));//10
```

函数

- 对箭头函数参数设置默认值

```
var a = 10;
var b = 20;
var c = function(){
  console.log('a:',a,'b:',b);
}
c();//a: 10 b: 20
var d = function(){
  console.log('a:',a,'b:',b);
}
d();//a: 10 b: 20
```

- 箭头函数

```
var fun1 = (x,y) => {
console.log(x,y);
console.log(x+y);
}
fun1(10,20);
```

  - 必须与函数结合

```
var fun1 = (x,y)=>{
console.log(x,y);
console.log(x+y);
}
var fun2 = (x,y)=>{
console.log(x,y);
console.log(x+y);
}
fun1(10,20);
fun2(30,40);
```

  - 实例

```
var fun1 = (x,y)=>{
console.log(x,y);
console.log(x+y);
}
function fun2(x,y){
console.log(x,y);
console.log(x+y);
}
fun1(10,20);
fun2(30,40);
```

ES6中的模块

- ES6为Map数据结构定义default引用 =>
  - 1 语 非模型外，Map数据结构default引用
  - 2 语 非模型外，模块导出数据Map数据结构
  - 3 语 模块模块中，Map数据结构数据的引用
  - 4 语 Map数据模 5数据结构 下每型之间

```
var map = new Map();
map.set('name', '张三');
console.log('name:',map.get('name'));//name: 张三

function hello(){
console.log('hello');
}
var map = new Map();
map.set('hello',hello);
var fun = map.get('hello');
fun();//hello
```

关于数组的展开运算

```
var arr = [1,2,3,4];
console.log(...arr);//1 2 3 4

function hello(x,y,z){
console.log(x,y,z);
}
var arr = [1,2,3];
hello(...arr);//1 2 3
```

String中新添加的方法

- includes("字符",startIndex)
  - 指示字符串字符串是否在主要字符串
  - 是否包含true 否 false 区分大小写

```
var str = 'abcdefg';
console.log( str.includes("a",1));//false
console.log( str.includes("a",0));//true
```

面向对象

- 构造 =>

```
var obj = {
name:"张三",
age:"18",
sex:"男",
}
var obj1 = {
name:"李四",
age:"19",
sex:"女",
}
var obj2 = {
name:"王五",
age:"20",
sex:"男",
}
var arr = [obj,obj1,obj2];
console.log(arr);
```

- 实例

```
class Person {
// 构造函数，一个类只能有一个
constructor(name,age,sex,profession){
this.name = name;
this.age = age;
this.sex = sex;
this.profession = profession;
}
// 实例方法
sell(){console.log("出售商品");}
eat(){console.log("吃饭");}
drink(){console.log("喝水");}
}

var zs = new Person("张三",18,"男","工程师");
console.log(zs);
```

面向对象的继承

- 对象的继承 =>
  - 实现继承
    - 构造函数的继承 =>

```
function Animal(){
this.eat = eat;
this.sleep = sleep;
}
Animal.prototype.run = function(){console.log("跑步了")}
var dog = new Animal();//实例化对象
var bird = new Animal();
function eat(){
console.log("吃饭");
}
function sleep(){console.log("睡觉了");}
var dog = new Animal();
dog.run();//跑步了
dog.eat();//吃饭
dog.sleep();//睡觉了
var bird = new Animal();
bird.run();//跑步了
bird.eat();//吃饭
bird.sleep();//睡觉了
```

    - 原型链的继承 =>

```
function Animal(){
this.eat = eat;
this.sleep = sleep;
}
Animal.prototype.run = function(){
console.log("跑步了")
}
function eat(){
console.log("吃饭");
}
function sleep(){
console.log("睡觉了");
}
var dog = new Animal();//实例化对象
var bird = new Animal();
dog.run();//跑步了
bird.run();//跑步了
```

    - 组合继承的继承 =>

```
function Animal(){
this.eat = eat;
this.sleep = sleep;
}
Animal.prototype.run = function(){
console.log("跑步了")
}
function eat(){
console.log("吃饭");
}
function sleep(){
console.log("睡觉了");
}
var dog = new Animal();//实例化对象
var bird = new Animal();
dog.run();//跑步了
bird.run();//跑步了
```

  - ES6的继承
    - 1.子类继承父类的属性和方法，但不能继承父类的构造函数，构造函数是每个类所独有的
    - 2.在子类的构造函数中，必须先调用super()，才可以引用this关键字，否则会报错。这是因为子类没有自己的this对象，而是继承父类的this对象，然后对其进行加工，而super()调用的是父类的构造函数。只有调用super()之后，才可以引用this关键字，因为子类的this对象是通过父类的this对象继承而来的，只有调用super()之后，才能拿到父类的this对象。

```
class Animal {
constructor(name,age,sex){
this.name = name;
this.age = age;
this.sex = sex;
}
run(){
console.log("跑步了")
}
}
class Dog extends Animal {
constructor(name,age,sex,color){
super(name,age,sex);//调用父类的构造函数，这是必须的规则
this.color = color;
}
run(){
console.log("跑步了")
}
bark(){
console.log("汪汪叫");
}
}
var d = new Dog("哈士奇",2,"男","白色");
d.run();//跑步了
d.bark();//汪汪叫
```

    - 1.子类继承父类的属性和方法，但不能继承父类的构造函数，构造函数是每个类所独有的
    - 2.在子类的构造函数中，必须先调用super()，才可以引用this关键字，否则会报错。这是因为子类没有自己的this对象，而是继承父类的this对象，然后对其进行加工，而super()调用的是父类的构造函数。只有调用super()之后，才可以引用this关键字，因为子类的this对象是通过父类的this对象继承而来的，只有调用super()之后，才能拿到父类的this对象。

```
class Animal {
constructor(name,age,sex){
this.name = name;
this.age = age;
this.sex = sex;
}
run(){
console.log("跑步了")
}
}
class Dog extends Animal {
constructor(name,age,sex,color){
super(name,age,sex);//调用父类的构造函数，这是必须的规则
this.color = color;
}
run(){
console.log("跑步了")
}
bark(){
console.log("汪汪叫");
}
}
var d = new Dog("哈士奇",2,"男","白色");
d.run();//跑步了
d.bark();//汪汪叫
```

中心主题

几种集合的总结

- Array
  - 特点：储存的元素可以是任何类型，并且是有序可以更改的
  - 创建方式
    - 构造函数：var arr=new Array()
    - 字面量语法：var arr=[]
  - 增：push
  - 删：splice
  - 改：splice
  - 查：indexOf
- Set（了解）
  - 特点：储存的元素可以是原型和对象类型，有序不可以更改
  - 创建方式：构造函数
  - 增：add
  - 删：delete
  - 长度：size
  - 改和查专用方法：没有
- Map
  - 特点：是一种键值对映射关系的集合
  - 创建方式：var map=new Map([["name",'小明'],["age",18],["sex",'男']])
  - 增：set
  - 删：delete
  - 改：set
  - 查：get

```
//三个基本遍历方法
1.for循环
2.for in
3.forEach(function(){})
//四个数组专用遍历方法
1.for循环
2.for in
2.for of
//四个通用对象遍历
1.for in
//四个数组Set
1.for of
2.forEach(function(){})
//四个对象Map
1.for of
2.forEach(function(){})
```

几种遍历集合的方式

- 传统for循环
  - 可以遍历
    - 字符串
      - 字面量
      - 变量

```
var str="hello";
for(var i = 0; i < str.length ;i++){
  console.log(str[i]+"--ASCII:"+str.charCodeAt(i));
}
```

    - 数组
      - 字面量
      - 变量

```
var arr=[1,2,3];
for(var i = 0; i < arr.length; i++){
  console.log(arr[i]);
}
```

  - 不可以遍历
    - 对象
      - 可以遍历map，但是改用自己，获取不到键个数
      - 可以遍历Map，但是改用自己，获取不到键个数
- for in
  - 可以遍历
    - 字符串

```
var str="hello";
for(var idx in str){
  console.log(str[idx]+"--"+str.charAt(idx))
}
```

    - 数组
      - 字面量

```
var arr=[1,2,3];
for(var idx in arr){
  console.log(arr[idx])
}
```

      - 变量

```
var arr=[1,2,3];
var idx=0;
for(idx in arr){
  console.log(arr[idx])
}
```

    - 对象

```
var obj={a:1,b:2,c:3};
for(var prop in obj ){
  console.log(obj[prop]);
}
```

  - 不可以遍历
    - map
    - set
- for of
  - 可以遍历
    - 字符串

```
var str="hello";
for(var chr of str){
  console.log(chr)
}
```

    - 数组

```
var arr=[1,2,3];
for(var s of arr){
  console.log(s);
  console.log(s[0]+":"+s[1])
}
```

    - 对象
    - set

```
var set=new Set([1,2,3]);
for(var s of set){
  console.log(s)
}
```

    - 数组
  - 不可以遍历
    - 对象
      - 可以遍历对象，需要实现Symbol.iterator接口
- forEach()
  - 可以遍历
    - 数组
      - 数组有数组的对象，可以有3个参数

```
arr.forEach(function(ele,index,arr){
  console.log(ele+":"+index);
  console.log(arr[index])
});
```

      - 数组Set的对象，可以有2个参数

```
set.forEach(function (ele,ele) {
  console.log(ele+"--"+ele2);
})
```

    - set
    - map
      - 数组Map的对象，可以有2个参数

```
map.forEach(function (value,key) {
  console.log(key+"--"+value);
})
```

  - 不可以遍历
    - 对象
    - 字符串

遍历的原理是Symbol.iterator，可以让数组Map set遍历

如果要给对象，任何类型，的数据结构（Map Set Array）部署遍历接口以及给它的值（任何类型），可以调用for of进行遍历

ES6迭代器

for of 循环内部的实现

```
//每个可迭代的Symbol.iterator方法
String.prototype[Symbol.iterator]=function(){
  var index = 0;
  var this=this;
  return{
    next:function(){
      if(index < this.length){
        return {value: this[index++],
        done:index > _this.length ? false : true
      }
    }
  }
}
for (var chr of str){
  console.log(chr)
}
```

对象的指针总结

```
//ES7的Array.prototype.includes(searchElement)
//检查数组是否包含某个元素，返回true或false
var arr=[1,2,3,4,5];
console.log(arr.includes(3));
//ES8的Object.values(obj)
//获取对象所有可枚举属性的值，返回一个数组
var obj={a:1,b:2,c:3};
console.log(Object.values(obj));
//ES8的Object.entries(obj)
//将对象转换为键值对数组，每个元素是[key, value]
var obj={name:'Alice',age:25};
console.log(Object.entries(obj));
//ES8的Object.fromEntries(iterable)
//将键值对数组转换回对象
var entries=[['name','Bob'],['age',30]];
console.log(Object.fromEntries(entries));
//ES9的Promise.prototype.finally(onFinally)
//无论Promise成功还是失败，都会执行
var p=Promise.resolve('success');
p.finally(function(){
  console.log('清理资源');
});
//ES9的Array.prototype.flatten(depth)
//将嵌套数组扁平化，depth默认为1
var nested=[1,[2,[3,[4]]]];
console.log(nested.flatten());
console.log(nested.flatten(2));
console.log(nested.flatten(Infinity));
//ES9的Array.prototype.flatMap(callbackFn)
//先map再flatten，只展开一层
var arr=[1,2,3];
console.log(arr.flatMap(function(x){
  return [x,x*2];
});
//ES9
```