🡪

使用$. post对后台进行Ajax请求，如果报错的信息为：TypeError: 'stepUp' called on an object that does not implement interface HTMLInputElement这样表示，在$. post给后台传入的数据字典中的数据，含有非普通数据，通常传入的是一个Object所以不行，我们要获取这个object的值，值为字符串或者数字等才可以传入给后台。

例如要传递给后台的数据为：

let data = {“A”: “B”, “C”:” D”};

这样我们只需要直接$.post (“”, data, …); 便可以。 Data本身会自动被Jquery进行序列化。

例如要传递给后台的数据为：

let data = {“A”: “B”, “C”: [“A”, “B”, “C”]};

这样我们Jquery的不会为data[“C”]下的数组进行序列化。我们需要变为

let data = {“A”:”B”, “C”: JSON.stringify([“A”,”B”, “C”])});才可以。

🡪

jquery的parent找寻的是父类元素，parents找寻的是祖先元素，不仅限于父类元素。

在找寻父类或者祖先元素节点的时候，还可以使用过滤器。过滤器有两种写法：

例如:$(” a”). parents(). filter (” div”) 也可以为$(”a”).parents(”div”)。

🡪

Jquery 的prop方法：

prop() 方法设置或返回被选元素的属性和值。

当该方法用于返回属性值时，则返回第一个匹配元素的值。

当该方法用于设置属性值时，则为匹配元素集合设置一个或多个属性/值对

🡪

HTML元素的contenteditable 属性，用于表达一个元素是否可以被编辑。如果为True, 那么该元素就变成了可编辑的元素，就如同textarea等输入元素 。

🡪

我们使用console.log可以从控制台输出中看到选中数据的类型。如果是jQuery则会显示是object, 并且原型是jQuery Object, 如果是原生DOM对象则会显示HTMLDOMELEMENT。

例如

**var** currentTD = $(**this**).parents('tr').find('td');

输出currentTD则是

Object { 0: td.setupId, 1: td.setupDes, 2: td.description, 3: td.network, 4: td.resources, 5: td.NFMPBranch, 6: td.NFMPBuildLoad, 7: td.NFMPType, 8: td.NSPVersion, 9: td.NSPLoad, … }

证明currentTD是一个jQuery对象，而不是一个简单的数组，可以继续使用jQuery的链式调用和隐形迭代。

再如

$.each(currentTD,**function** (index,element){  
 console.log(element)  
 console.log($(element))

输出为

<td class="setupId" contenteditable="false"> -> 证明如果从jQuery对象中使用each方法取出时，每一个元素将会是原生的DOM对象。

Object [ td.setupId ] -> 从而使用$进行构建，将成为对应DOM对象的jQuery对象。

🡪

jQuery有两种each遍历方法

$(selector). each (function (index, element)) -> 多用于遍历DOM对象

$. each (data, function (index, element) {}) -> 多用于遍历数组, 当然数组中也可以是DOM对象

🡪

JavaScript中undefined与null区别

null是一个表示” 无” 的对象，转为数值时为0，是用来存放一个空的对象的，也就是被声明而且已经初始化了。表示” 没有对象” ，即该处不应该有值。

undefinedd表示” 缺少值” ，就是此处应该有一个值，但是还没有定义。是一个对象被声明但未初始化的（未定义的）。转为数值时为NaN。

🡪

Jquery添加元素的方式：

append() - 在被选元素的结尾，在被选定元素内插入内容

prepend() - 在被选元素的开头，在被选定元素内插入内容

after() - 在被选元素之后，在被选定元素外插入内容

before() - 在被选元素之前，在被选定元素外插入内容

在括号中可以插入以HTML形式创建的新元素，以Jquery创建的新元素，以DOM创建的新元素

var txt1="<p>Text. </p> “; // 以 HTML 创建新元素

var txt2=$("<p></p>").text("Text."); // 以 jQuery 创建新元素

var txt3=document.createElement("p"); // 以 DOM 创建新元素

🡪

Jquery中filter () 方法与find () 方法的区别。

Find方法是选择器所选择的元素的里面，他们的儿子元素中返回合符条件的元素。

Filter方法是从选自其所选择的元素中（非儿子元素中），找寻其中合符条件的元素。

🡪

Jquery使用Data对象来获取时间信息。例如：

var mydate = new Date();  
mydate.getYear(); //获取当前年份(2位)  
mydate.getFullYear(); //获取完整的年份(4位,1970-????)  
mydate.getMonth(); //获取当前月份(0-11,0代表1月)  
mydate.getDate(); //获取当前日(1-31)  
mydate.getDay(); //获取当前星期X(0-6,0代表星期天)  
mydate.getTime(); //获取当前时间(从1970.1.1开始的毫秒数)  
mydate.getHours(); //获取当前小时数(0-23)  
mydate.getMinutes(); //获取当前分钟数(0-59)  
mydate.getSeconds(); //获取当前秒数(0-59)

🡪

# Jquery的控件jQuery Validate是一个Jquery的插件，主要用于增强表单的验证。导入之后可以使用form object的validate（）方法进行验证表单。

🡪

Jquery在捕获id属性中带有.的需要用到转义符号，所以最后设置id的时候，id的值为纯数字

🡪

在高版本的Jquery中，引入了prop方法，其中prop方法与attr方法的区别在于

对于HTML元素本身就带有的固有属性，在处理时，使用prop方法。

对于HTML元素我们自己自定义的DOM属性，在处理时，使用attr方法。

🡪

Jquery选择器选择的时候，表示与条件用: 表示或条件用,

例如：

$('div[name="a"],div[name="b"]') //选择div属性为A或者div属性为b的那些元素

has(td[col="0"]):has(td[col="1"]) //选择同时含有td属性值为0并且含有td属性值为1的元素

🡪

jQuery主要版本有1.x,2.x,3. x.3代版本是最后一代jQuery版本，现在jQuery已经停止开发 （GitHub已经完全不使用任何jQuery代码，bootstrap5 也将不使用任何jQuery代码），但还是主流前端JS插件库，还有不少网站采用jQuery. 其被替代的原因是现代Web开发采用面向数据开发，例如React, Vue等。并且浏览器兼容问题得到统一， jQuery的选择器也由于原生Javascript的选择器进步而弱化。

🡪

Jquery选择器返回的永远是一个Jquery对象， 若未找到指定元素， 就会返回一个空的Jquery对象。这个对象类似为数组，支持链式操作

通常，我们用javascript判断DOM是否存在的时候可以使用：

if(document.getElementById(“wrap”) //如果不存在这个DOM则为false

但是在Jquery中就不能这样使用来判断DOM是否存在:

if (Jquery(‘#wrap’)

因为Jquery选择器返回的永远是一个数组对象， 若未能找到指定元素，就会返回一个空的数组对象，这样还是判定为True.

所以应该为if (Jquery(‘#wrap’).length > 1)

🡪

Jquery祖先遍历：

|  |  |
| --- | --- |
| parent () | parent () 方法返回被选元素的直接父元素。 |
| parents () | parents () 方法返回被选元素的所有祖先元素，它一路向上直到文档的根元素 |
| parentsUntil() | parentsUntil() 方法返回介于两个给定元素之间的所有祖先元素 |
| closest () | closest () 方法获得匹配选择器的第一个祖先元素，从当前元素开始沿 DOM 树向上 |

🡪

<script>标签可以使用在<head>以及<body>当中。

一般将需要从外部通过<script src=”…”>方式引入的js文件放置到<head>当中。并且由于浏览器是从上到下解析HTML的。放在head里的js代码，会在body解析之前,DOM构建之前被解析。所以这些js文件的源码当中，经常将代码放在document.ready()当中（或者简易写法$(function(){})），从而确保页面加载完成后才进行执行。

<body>中的js一般放置一些该页面的生成内容代码，或者一些特殊绑定。同样<body>中的js会随着浏览器是从上到下解析执行。

例如：

<html>  
 <body>  
 <h3 id="A">A</h3>  
 <h3 id="B">B</h3>  
 <script>

document.getElementById("B").innerHTML="GGGG"; //有效，此时id=”B”元素已经生成   
 document.getElementById("C").innerHTML="GGGG"; //无效， 此时id=”C”元素还没生成  
 </script>  
 <h3 id="C">C</h3>  
 <h3 id="D">D</h3>  
 </body>  
</html>

同理，为了想确保<body>中的js代码在DOM元素构建以后才执行，也会将js源码放入document.ready()当中（或者简易写法$(function(){})）。

🡪