基于 jq 的纯 js 版 json 配置工具开发

——————2019.7.30 花名：韵方

首先来看一下成品吧，没有写 css 略微有点小丑啊不要介意哈：

然后直接上代码哈，代码其实一共可以分为以下的这么几步完成。1.编写 html 组件（包括目前看到的所有按钮和 text 框）。2.编写+-号的逻辑，包括自动生成编号和文本框以及他们的 id。3.编写生成 json 按钮的代码。此处分为 3 小步。（1）将原始字符串拆分成两个队列，一个队列存放文本内容，另一个队列存放配置字符数字（2）将要配置的字符逐行存入二维数组中，二维数组每行为一个配置，每列为配置的一行数据（3）合并字符串并输出在对应的位置上。花里胡哨的就不多说了，直接上代码。

首先是 html 模块的代码，此处不过多解释了，就是一些很基础的组件和属性。

```
<!DOCTYPE html>
<html>
<head>
  <title>json 生成</title>
  <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
</head>
<body>
  <p>配置规则（配置数据用<1>,<2>...代替）</p>
  <textarea rows="10" cols="100" placeholder="输入配置字符串" id="yuan"></textarea>
  <p>输入配置</p>
  <div id="div1">
    <div id="zif1">
      <p>(1)</p>
      <textarea placeholder="输入要配置的字符" id="1"></textarea>
    </div>
  </div>

    </br>
      <button id="add">+</button>
      <button id="sub">-</button>
  </br>

  <button id="get_json">生成json</button>
  </br></br>
  <textarea rows="10" cols="100" value="ss" placeholder="生成的json在这里复制" id="res"></textarea>

</body>
```

然后是加减配置字符 div 模块的代码，这里要注意配置 textarea 的 id，因为下面会用到哦。有加模块和减模块两个模块。

```
$(document).ready(function(){
  var n = 1;//全局变量 变量数量
  //加模块代码
    $("#add").click(function(){
      n++;
      var str = "";
      str += "<div id=zif"+n+">";
      str += "<p>("+n+")</p>";
    str += "<textarea placeholder=输入要配置的字符 id="+n+"></textarea>";
    str += "</div>"
    $("#div1").append(str);
    });
      //减模块代码
    $("#sub").click(function(){
      if(n == 1){
        return;
      }
      $("#zif"+(n)+"").remove();
      n--;
    });
```

然后就是对原始字符串进行解析，使当他读到<n>时会自动去匹配配置好的字符串，然后进行一一的对应，替换成我们最终要的字符。这里我使用了两个队列的数据结构，一个队列存放无<n>的字符串，另外一个队列存放 n 的值，以便为我们去下面的配置好的字符那里去匹配。

这里要提一下 js 入队列出队列的基本语法。

.unshift()//入队列

.pop()//出队列

```
$("#get_json").click(function(){
  //将原始字符串拆分成两个队列，一个队列存放文本内容，另一个队列存放配置字符数字
  var post_str_old = $("#yuan").val();
  var dui = [];
  var index_first = 0;
  var number = [];
  var dui_len = 0;
  var number_len = 0;
  for(var i = 0;i < post_str_old.length;i++){
    str_one = post_str_old.substring(i,i+1);
    if(str_one == "<"){
      dui.unshift(post_str_old.substring(index_first,i));
      dui_len++;
      str_x = "";
      i++;
      if(post_str_old.substring(i,i+1) != ">"){
        str_x += post_str_old.substring(i,i+1);
        i++;
      }
              number.unshift(str_x);
              number_len++;
      index_first = i+1;
    }
  }
  dui.unshift(post_str_old.substring(index_first,i));
  dui_len++;
```

再然后就是将需要配置的字符存起来，以便与去匹配需要匹配的字符了。这里我用了二维数组的形式进行储存。

```
//将要配置的字符逐行存入二维数组中，二维数组每行为一个配置，每列为配置的一行数据
var hang = 0;
var arr = new Array();
for(var i = 0;i < n;i++){
  // var str = "\"#"+(i+1)+"\"";
  // alert(str);
  //var yuan = $("\"#"+(i+1)+"\"").val();
  var yuan = $("#"+(i+1)+"").val();
  //alert(yuan);
  arr[i] = new Array();
  arr[i] = yuan.split("\n");
  hang = arr[i].length;
  // for(var g = 0;g < arr[i].length;g++){
  //  alert(arr[i][g]);
  // }
}
```

最后就是合并单元格的操作了，将队列拿来交叉一一出队列，其中一个数字的队列去匹配二维数组中的数据，逐行逐行重复执行，从而得到最后的结果。

```
        //合并字符串模块
        var number_tem = [];
        var dui_tem  = [];
        var res = "";
    for(var i = 0;i < hang;i++){
      dui_tem = qiang_copy(dui,dui_tem,dui_len);
      number_tem = qiang_copy(number,number_tem,number_len);
      var str_res = "";
      var dui_tem_one = dui_tem.pop();
      console.log(dui_tem_one);
      while(dui_tem_one != null){
        str_res += dui_tem_one;
        number_tem_one = number_tem.pop();
        if (number_tem_one != null) {
          var number_index = parseInt(number_tem_one);
          console.log(number_index);
          str_res += arr[number_index-1][i];
        }
        dui_tem_one = dui_tem.pop();
      }
      console.log(str_res);
      res += str_res+"\n";
    }
    $("#res").text(res);//输出
  });
```

注意了，在此时我碰到了一个问题，就是每次逐行重复执行的时候，需要对队列进行一次进出的操作，当然这可以用环状队列来实现，但我最初是用普通队列复制一个新的队列来实现，这时我发现只要改变新复制的队列，原先的队列也会发生改变。

经过查找相关的资料，我知道了原来 js 的队列使用的是地址引用，而我们一般采用=的形式赋值，只会对其地址进行复制，所以改了新的，原来的也就变了，所以，我们需要采取强复制的方式进行赋值并且进行出入的操作。

以下为强赋值的代码，中间经过了一层普通的值的引用，所以就不会只复制地址了。

```
//强复制队列模块
function qiang_copy(a,b,len){
  for(var i = 0;i < len;i++){
    var tem = a.pop();
    b.unshift(tem);
    a.unshift(tem);
  }
  return b;
}
```

最后，介于 es6 新语法的规范，上述代码中所有的 var 均可用 const 代替，谢谢。

知识点：js 弱复制与强复制的产生原因，区别和解决办法
Js 队列的基本操作