WEB 分页组件使用说明

\*\* 当前版本： Version 1.0.0

\*\* 更新时间： 2015年4月2日17:34:09

\*\* 更新人员： TuzK1ss

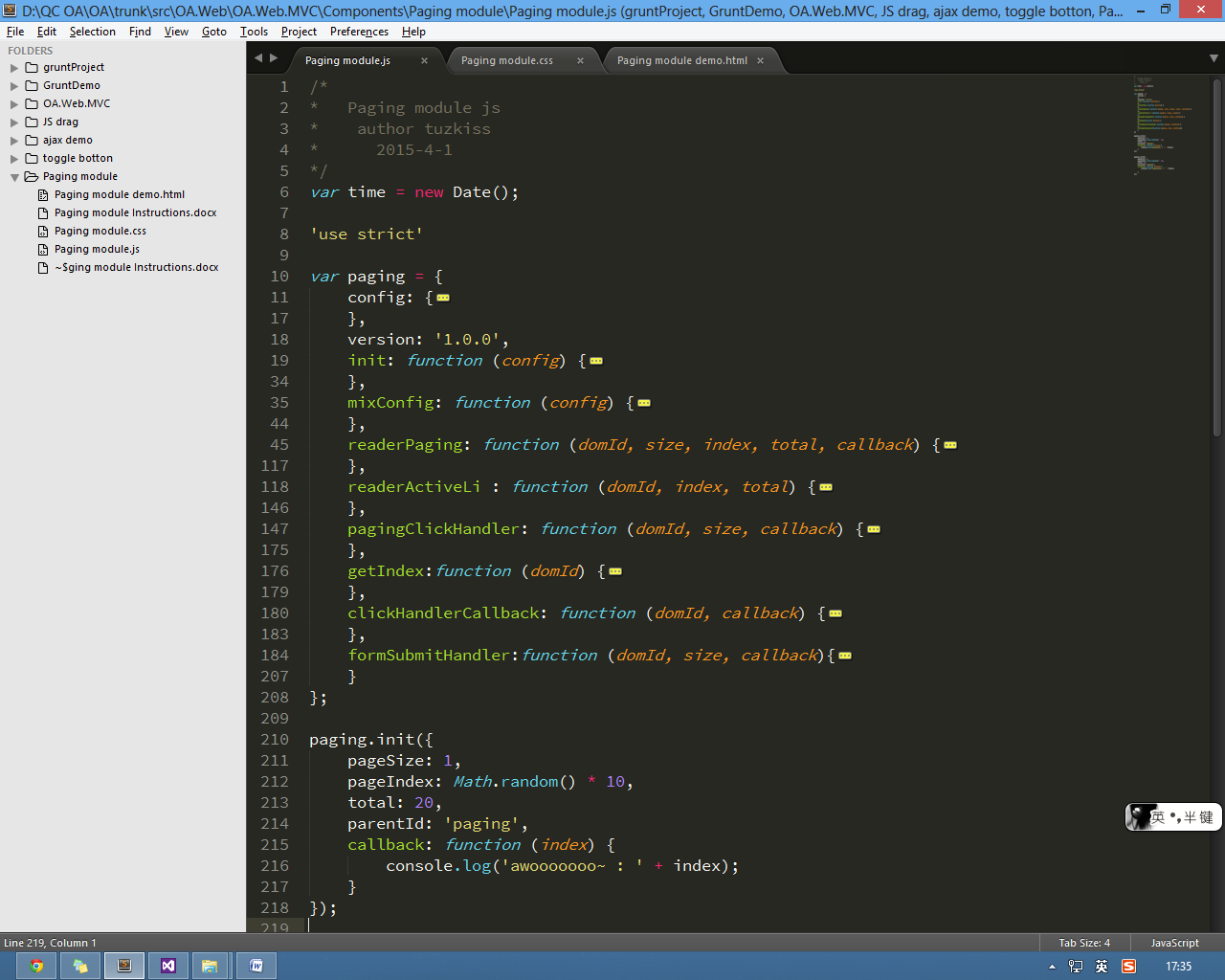
\*\* 使用范围： chrome， firefox， ie 10 ， ie9 （截止更新时间）

\*\* 尚未完成： 兼容ie8， 手动跳转回车跳转事件， 多样式选择

Version 1.0.0版本说明

截图如下：



主要定义代码如下：

**其中包括**

自定义的默认初始化config ， 版本号说明version，

合并传递config和默认config， 渲染分页UL方法readerPaging,

渲染当前页方法readerActiveLi, 分页中按钮点击事件pagingClickHandler，

获取当前页页码方法getIndex， 点击callback事件clickHandlerCallback，

点击确认按钮跳转事件formSubmitHandler。

**需要传递的参数包括**，

pageSize （每页多少条数据，可选，默认10条），

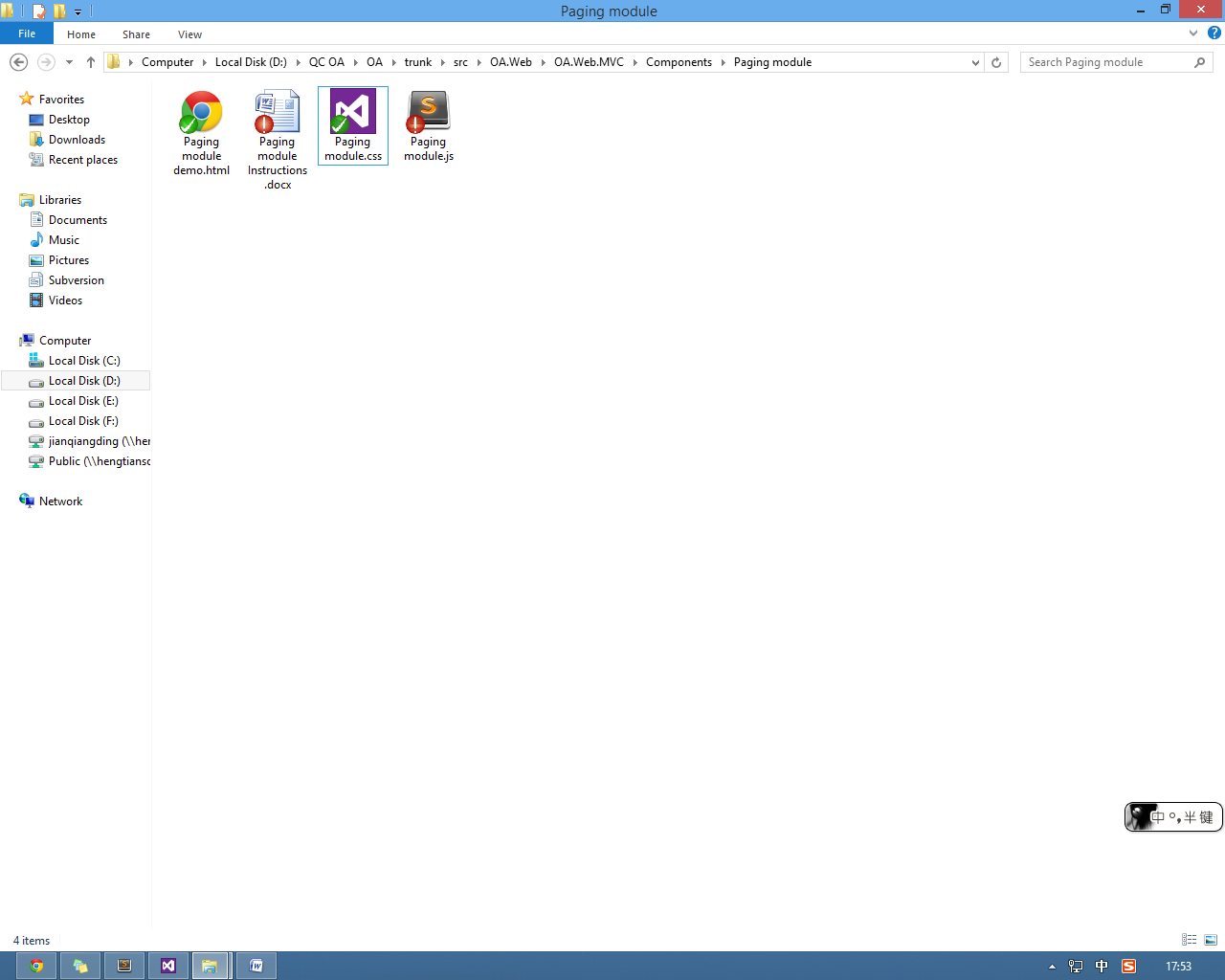
pageIndex （当前页页码， 可选，默认第1页），

total （总条数， 必选），

parentId （分页组件的父ID， 必选），

callback （点击返回函数， 必选， 传入参数index）；

**例如（调用方法）**：



引入上文中的JS和css文件到需要的页面，然后在调用HTML页面中调用如下：

paging.init({

pageSize: 10,

pageIndex: 1

total: 20,

parentId: 'paging',

callback: function (index) {

console.log('awooooooo~ : ' + index);

}

});

具体代码见下， 具体效果见[demo](Paging%20module%20demo.html)

具体代码如下： （为格式化）

1. **var** paging = {
2. config: {
3. pageSize: 1,
4. pageIndex: Math.random() \* 10,
5. total: 10,
6. parentId: 'document.body',
7. callback:**null**
8. },
9. version: '1.0.0',
10. init: **function** (config) {
12. config = **this**.mixConfig(config);
14. **var** domId = config.parentId,
15. size = config.pageSize,
16. index = config.pageIndex,
17. total = config.total,
18. callback = config.callback;
20. **this**.readerPaging(domId, size, index, total, callback)
21. .pagingClickHandler(domId, size, callback)
22. .formSubmitHandler(domId, size, callback);
24. **return** **this**;
25. },
26. mixConfig: **function** (config) {
28. **for** (**var** i **in** config) {
29. **this**.config[i] = config[i] || **this**.config[i];
30. }
32. **this**.config["pageIndex"] = **this**.config["pageIndex"] > **this**.config["total"] ? **this**.config["total"] :**this**.config["pageIndex"];
34. **return** **this**.config;
35. },
36. readerPaging: **function** (domId, size, index, total, callback) {
37. **var** dom = document.getElementById(domId),
38. totalPage = Math.ceil(total / size),
39. last = 0,
40. html = '';
41. dom.innerHTML = '';
43. html += '<ul class="paging-ul">';
45. html += '<li class="prev"><a data-size="prev" href="javascript:;">上一页</a></li>';
47. index = Math.ceil(index);
49. **if** (totalPage <= 7 ) {
50. // 当分页总页数小于7时全部显示
52. last = index > 5 ? index : 5;
54. **for** (**var** i = 1; i < last + 1; i ++) {
55. html += '<li class="item"><a data-size="' + i + '" href="javascript:;">' + i + '</a></li>';
56. }
58. } **else** **if** (totalPage > 7 && index <= 5) {
59. last = index + 2 < 5 ? 5 : index + 2;
61. **for** (**var** i = 1; i < last + 1; i ++) {
62. html += '<li class="item"><a data-size="' + i + '" href="javascript:;">' + i + '</a></li>';
63. }
65. html += '<li class="item dot">...</li>';
67. } **else** **if** (totalPage > 7 && index > 5)  {
68. // 当前页大于第五页
70. // 前两页 + ...
71. **for** (**var** i = 1; i < 3; i ++) {
72. html += '<li class="item"><a data-size="' + i + '" href="javascript:;">' + i + '</a></li>';
73. }
75. html += '<li class="item dot">...</li>';
77. // 后五页
78. **if** ( index + 2 < totalPage ) {
80. **for** (**var** i = index - 2; i < index + 3; i ++) {
81. html += '<li class="item"><a data-size="' + i + '" href="javascript:;">' + i + '</a></li>';
82. }
83. html += '<li class="item dot">...</li>';
85. } **else** {
87. **for** (**var** i = index - 2; i < totalPage + 1; i ++) {
88. html += '<li class="item"><a data-size="' + i + '" href="javascript:;">' + i + '</a></li>';
89. }
91. }
93. }
95. html += '<li class="next"><a data-size="next"  href="javascript:;">下一页</a></li>';
97. html += '<li class="paging-total">';
98. html += ('<span id="' + domId +'PageForm" class="form">共 <span id="' + domId +'Total">' + totalPage + '</span> 页, 到第 <input id="' + domId +'Index" type="number" value="' + index +'" min="1" max="' + total + '" class="paging-index" value="' + index + '"</input> </div>页 ' +
99. '<input id="' + domId +'Comfirm" class="go-paging" type="submit"  value="确定"></span>');
100. html += '</li></ul>';

103. dom.innerHTML = html;
105. **this**.readerActiveLi(domId, index, total).formSubmitHandler(domId, size, callback);
107. **return** **this**;
108. },
109. readerActiveLi : **function** (domId, index, total) {
110. **var** item = document.querySelectorAll('#' + domId + ' .item a'),
111. indexA,
112. tmp;
114. **if** (1 == index) {
115. tmp = document.querySelector('#' + domId + ' .prev a');
116. tmp.style.color = "#999";
117. tmp.setAttribute('disabled', 'true');
118. } **else** **if** (total == index) {
119. tmp = document.querySelector('#' + domId + ' .next a');
120. tmp.style.color = "#999";
121. tmp.setAttribute('disabled', 'true');
122. }
124. **for**(**var** i = 0, len = item.length; i < len; i ++) {
125. **var** indexA = item[i].getAttribute('data-size') || 0;
127. **if** (index == indexA) {
128. item[i].parentNode.className += ' active-li';
129. item[i].setAttribute('disabled', 'true');
130. **return** **this**;
131. }
132. }


136. **return** **this**;
137. },
138. pagingClickHandler: **function** (domId, size, callback) {
139. **var** that = **this**;
141. **var** paging = document.getElementById(domId);
142. paging.addEventListener('click', **function** (e){
143. e = e || window.event;
144. **var** target = e.target || e.srcElement;
146. //console.log(target);
147. **var** index = target.getAttribute('data-size'),
148. isDisabled = target.getAttribute('disabled'),
149. total = document.getElementById(domId + 'Total').innerHTML,
150. active = parseInt(document.querySelector('#' + domId +' .active-li a').getAttribute('data-size'));
152. **if** (index && !isDisabled) {
153. **if** (index === 'prev' ) {
154. that.readerPaging(domId, size, active - 1, total, callback);
155. } **else** **if** (index === 'next' ) {
156. that.readerPaging(domId, size, active + 1, total, callback);
157. } **else** {
158. that.readerPaging(domId, size, index, total, callback);
159. }
161. that.clickHandlerCallback(domId, callback);
162. }
163. }, **false**);
165. **return** **this**;
166. },
167. getIndex:**function** (domId) {
168. **var** index = parseInt(document.querySelector('#' + domId +' .active-li a').getAttribute('data-size'));
169. **return** index;
170. },
171. clickHandlerCallback: **function** (domId, callback) {
172. **var** index = **this**.getIndex(domId);
173. callback(index);
174. },
175. formSubmitHandler:**function** (domId, size, callback){
176. **var** that = **this**,
177. input = document.getElementById(domId + 'Comfirm');;
179. input.addEventListener('click', **function** (e){
180. e = e || window.event;
181. e.preventDefault();
183. **var** form = document.getElementById(domId + 'PageForm'),
184. total = parseInt(form.getElementsByTagName('span')[0].innerHTML),
185. index = parseInt(form.getElementsByTagName('input')[0].value),
186. active = that.getIndex(domId);
188. **if** (index !== active && index <= total && index > 0) {
189. that.readerPaging(domId, size, index, total, callback)
190. .clickHandlerCallback(domId, callback);
191. } **else** **if** (index !== active) {
192. form.getElementsByTagName('input')[0].value = 1;
193. }
195. }, **false**);
197. **return** **this**;
198. }
199. };