**[javascript]** [view plain](http://blog.csdn.net/ly1577592799/article/details/43409053" \o "view plain) [copy](http://blog.csdn.net/ly1577592799/article/details/43409053" \o "copy)

1. /\*
2. \* 取得字符串的字节长度
3. \*/
4. **function** strlen(str) {
5. **var** i;
6. **var** len;
8. len = 0;
9. **for** (i = 0; i < str.length; i++) {
10. **if** (str.charCodeAt(i) > 255) len += 2; **else** len++;
11. }
12. **return** len;
13. }

16. **function** strlen(str) {
17. **var** i;
18. **var** len;
20. len = 0;
21. **for** (i = 0; i < str.length; i++) {
22. **if** (str.charCodeAt(i) > 255) len += 2; **else** len++;
23. }
24. **return** len;
25. }
27. /\*
28. \* 判断是否为数字，是则返回true,否则返回false
29. \*/
30. **function** f\_check\_number(obj) {
31. **if** (/^\d+$/.test(obj.value)) {
32. **return** **true**;
33. }
34. **else** {
35. f\_alert(obj, "请输入数字");
36. **return** **false**;
37. }
38. }

41. **function** f\_check\_number(obj) {
42. **if** (/^\d+$/.test(obj.value)) {
43. **return** **true**;
44. }
45. **else** {
46. f\_alert(obj, "请输入数字");
47. **return** **false**;
48. }
49. }
51. /\*
52. \* 判断是否为自然数，是则返回true,否则返回false
53. \*/
54. **function** f\_check\_naturalnumber(obj) {
55. **var** s = obj.value;
56. **if** (/^[0-9]+$/.test(s) && (s > 0)) {
57. **return** **true**;
58. }
59. **else** {
60. f\_alert(obj, "请输入自然数");
61. **return** **false**;
62. }
63. }

66. **function** f\_check\_naturalnumber(obj) {
67. **var** s = obj.value;
68. **if** (/^[0-9]+$/.test(s) && (s > 0)) {
69. **return** **true**;
70. }
71. **else** {
72. f\_alert(obj, "请输入自然数");
73. **return** **false**;
74. }
75. }

78. /\*
79. \* 判断是否为整数，是则返回true,否则返回false
80. \*/
81. **function** f\_check\_integer(obj) {
82. **if** (/^(\+|-)?\d+$/.test(obj.value)) {
83. **return** **true**;
84. }
85. **else** {
86. f\_alert(obj, "请输入整数");
87. **return** **false**;
88. }
89. }

92. **function** f\_check\_integer(obj) {
93. **if** (/^(\+|-)?\d+$/.test(obj.value)) {
94. **return** **true**;
95. }
96. **else** {
97. f\_alert(obj, "请输入整数");
98. **return** **false**;
99. }
100. }
102. /\*
103. \* 判断是否为实数，是则返回true,否则返回false
104. \*/
106. **function** f\_check\_float(obj) {
107. **if** (/^(\+|-)?\d+($|\.\d+$)/.test(obj.value)) {
108. **return** **true**;
109. }
110. **else** {
111. f\_alert(obj, "请输入实数");
112. **return** **false**;
113. }
114. }

117. **function** f\_check\_float(obj) {
118. **if** (/^(\+|-)?\d+($|\.\d+$)/.test(obj.value)) {
119. **return** **true**;
120. }
121. **else** {
122. f\_alert(obj, "请输入实数");
123. **return** **false**;
124. }
125. }
127. /\*
128. \* 校验数字的长度和精度
129. \*/
131. **function** f\_check\_double(obj) {
132. **var** numReg;
133. **var** value = obj.value;
134. **var** strValueTemp, strInt, strDec;
135. **var** dtype = obj.eos\_datatype;
136. **var** pos\_dtype = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")).split(",");
137. **var** len = pos\_dtype[0], prec = pos\_dtype[1];
138. **try** {
139. numReg = /[\-]/;
140. strValueTemp = value.replace(numReg, "");
141. numReg = /[\+]/;
142. strValueTemp = strValueTemp.replace(numReg, "");
143. //整数
144. **if** (prec == 0) {
145. numReg = /[\.]/;
146. **if** (numReg.test(value) == **true**) {
147. f\_alert(obj, "输入必须为整数类型");
148. **return** **false**;
149. }
150. }
151. **if** (strValueTemp.indexOf(".") < 0) {
152. **if** (strValueTemp.length > (len - prec)) {
153. f\_alert(obj, "整数位不能超过" + (len - prec) + "位");
154. **return** **false**;
155. }
156. } **else** {
157. strInt = strValueTemp.substr(0, strValueTemp.indexOf("."));
158. **if** (strInt.length > (len - prec)) {
159. f\_alert(obj, "整数位不能超过" + (len - prec) + "位");
160. **return** **false**;
161. }
162. strDec = strValueTemp.substr((strValueTemp.indexOf(".") + 1), strValueTemp.length);
163. **if** (strDec.length > prec) {
164. f\_alert(obj, "小数位不能超过" + prec + "位");
165. **return** **false**;
166. }
167. }
168. **return** **true**;
169. } **catch** (e) {
170. alert("in f\_check\_double = " + e);
171. **return** **false**;
172. }
173. }

176. **function** f\_check\_double(obj) {
177. **var** numReg;
178. **var** value = obj.value;
179. **var** strValueTemp, strInt, strDec;
180. **var** dtype = obj.eos\_datatype;
181. **var** pos\_dtype = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")).split(",");
182. **var** len = pos\_dtype[0], prec = pos\_dtype[1];
183. **try** {
184. numReg = /[\-]/;
185. strValueTemp = value.replace(numReg, "");
186. numReg = /[\+]/;
187. strValueTemp = strValueTemp.replace(numReg, "");
188. //整数
189. **if** (prec == 0) {
190. numReg = /[\.]/;
191. **if** (numReg.test(value) == **true**) {
192. f\_alert(obj, "输入必须为整数类型");
193. **return** **false**;
194. }
195. }
196. **if** (strValueTemp.indexOf(".") < 0) {
197. **if** (strValueTemp.length > (len - prec)) {
198. f\_alert(obj, "整数位不能超过" + (len - prec) + "位");
199. **return** **false**;
200. }
201. } **else** {
202. strInt = strValueTemp.substr(0, strValueTemp.indexOf("."));
203. **if** (strInt.length > (len - prec)) {
204. f\_alert(obj, "整数位不能超过" + (len - prec) + "位");
205. **return** **false**;
206. }
207. strDec = strValueTemp.substr((strValueTemp.indexOf(".") + 1), strValueTemp.length);
208. **if** (strDec.length > prec) {
209. f\_alert(obj, "小数位不能超过" + prec + "位");
210. **return** **false**;
211. }
212. }
213. **return** **true**;
214. } **catch** (e) {
215. alert("in f\_check\_double = " + e);
216. **return** **false**;
217. }
218. }
220. /\*
221. \* 校验数字的最小最大值
222. \* 返回bool
223. \*/
225. **function** f\_check\_interval(obj) {
226. **var** value = parseFloat(obj.value);
228. **var** dtype = obj.eos\_datatype;
229. **var** pos\_dtype = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")).split(",");
231. **var** minLimit = pos\_dtype[0];
232. **var** maxLimit = pos\_dtype[1];
233. **var** minVal = parseFloat(pos\_dtype[0]);
234. **var** maxVal = parseFloat(pos\_dtype[1]);
236. **if** (isNaN(value)) {
237. f\_alert(obj, "值必须为数字");
238. **return** **false**;
239. }
240. **if** ((isNaN(minVal) && (minLimit != "-")) || (isNaN(maxVal) && (maxLimit != "+"))) {
241. f\_alert(obj, "边界值必须为数字或-、+");
242. **return** **false**;
243. }
245. **if** (minLimit == "-" && !isNaN(maxVal)) {
246. **if** (value > maxVal) {
247. f\_alert(obj, "值不能超过" + maxVal);
248. **return** **false**;
249. }
250. }
252. **if** (!isNaN(minVal) && maxLimit == "+") {
253. **if** (value < minVal) {
254. f\_alert(obj, "值不能小于" + minVal);
255. **return** **false**;
256. }
257. }
259. **if** (!isNaN(minVal) && !isNaN(maxVal)) {
260. **if** (minVal > maxVal) {
261. f\_alert(obj, "起始值" + minVal + "不能大于终止值" + maxVal);
262. } **else** {
263. **if** (!(value <= maxVal && value >= minVal)) {
264. f\_alert(obj, "值应该在" + minVal + "和" + maxVal + "之间");
265. **return** **false**;
266. }
267. }
268. }
269. **return** **true**;
270. }

273. **function** f\_check\_interval(obj) {
274. **var** value = parseFloat(obj.value);
276. **var** dtype = obj.eos\_datatype;
277. **var** pos\_dtype = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")).split(",");
279. **var** minLimit = pos\_dtype[0];
280. **var** maxLimit = pos\_dtype[1];
281. **var** minVal = parseFloat(pos\_dtype[0]);
282. **var** maxVal = parseFloat(pos\_dtype[1]);
284. **if** (isNaN(value)) {
285. f\_alert(obj, "值必须为数字");
286. **return** **false**;
287. }
288. **if** ((isNaN(minVal) && (minLimit != "-")) || (isNaN(maxVal) && (maxLimit != "+"))) {
289. f\_alert(obj, "边界值必须为数字或-、+");
290. **return** **false**;
291. }
293. **if** (minLimit == "-" && !isNaN(maxVal)) {
294. **if** (value > maxVal) {
295. f\_alert(obj, "值不能超过" + maxVal);
296. **return** **false**;
297. }
298. }
300. **if** (!isNaN(minVal) && maxLimit == "+") {
301. **if** (value < minVal) {
302. f\_alert(obj, "值不能小于" + minVal);
303. **return** **false**;
304. }
305. }
307. **if** (!isNaN(minVal) && !isNaN(maxVal)) {
308. **if** (minVal > maxVal) {
309. f\_alert(obj, "起始值" + minVal + "不能大于终止值" + maxVal);
310. } **else** {
311. **if** (!(value <= maxVal && value >= minVal)) {
312. f\_alert(obj, "值应该在" + minVal + "和" + maxVal + "之间");
313. **return** **false**;
314. }
315. }
316. }
317. **return** **true**;
318. }
320. /\*
321. 用途：检查输入字符串是否只由汉字组成
322. 如果通过验证返回true,否则返回false
323. \*/
325. **function** f\_check\_zh(obj) {
326. **if** (/^[\u4e00-\u9fa5]+$/.test(obj.value)) {
327. **return** **true**;
328. }
329. f\_alert(obj, "请输入汉字");
330. **return** **false**;
331. }

334. **function** f\_check\_zh(obj) {
335. **if** (/^[\u4e00-\u9fa5]+$/.test(obj.value)) {
336. **return** **true**;
337. }
338. f\_alert(obj, "请输入汉字");
339. **return** **false**;
340. }
342. /\*
343. \* 判断是否为小写英文字母，是则返回true,否则返回false
344. \*/
346. **function** f\_check\_lowercase(obj) {
347. **if** (/^[a-z]+$/.test(obj.value)) {
348. **return** **true**;
349. }
350. f\_alert(obj, "请输入小写英文字母");
351. **return** **false**;
352. }

355. **function** f\_check\_lowercase(obj) {
356. **if** (/^[a-z]+$/.test(obj.value)) {
357. **return** **true**;
358. }
359. f\_alert(obj, "请输入小写英文字母");
360. **return** **false**;
361. }
363. /\*
364. \* 判断是否为大写英文字母，是则返回true,否则返回false
365. \*/
367. **function** f\_check\_uppercase(obj) {
368. **if** (/^[A-Z]+$/.test(obj.value)) {
369. **return** **true**;
370. }
371. f\_alert(obj, "请输入大写英文字母");
372. **return** **false**;
373. }

376. **function** f\_check\_uppercase(obj) {
377. **if** (/^[A-Z]+$/.test(obj.value)) {
378. **return** **true**;
379. }
380. f\_alert(obj, "请输入大写英文字母");
381. **return** **false**;
382. }
384. /\*
385. \* 判断是否为英文字母，是则返回true,否则返回false
386. \*/
388. **function** f\_check\_letter(obj) {
389. **if** (/^[A-Za-z]+$/.test(obj.value)) {
390. **return** **true**;
391. }
392. f\_alert(obj, "请输入英文字母");
393. **return** **false**;
394. }

397. **function** f\_check\_letter(obj) {
398. **if** (/^[A-Za-z]+$/.test(obj.value)) {
399. **return** **true**;
400. }
401. f\_alert(obj, "请输入英文字母");
402. **return** **false**;
403. }
405. /\*
406. 用途：检查输入字符串是否只由汉字、字母、数字组成
407. 输入：
408. value：字符串
409. 返回：
410. 如果通过验证返回true,否则返回false
411. \*/
413. **function** f\_check\_ZhOrNumOrLett(obj) {    //判断是否是汉字、字母、数字组成
414. **var** regu = "^[0-9a-zA-Z\u4e00-\u9fa5]+$";
415. **var** re = **new** RegExp(regu);
416. **if** (re.test(obj.value)) {
417. **return** **true**;
418. }
419. f\_alert(obj, "请输入汉字、字母或数字");
420. **return** **false**;
421. }

424. **function** f\_check\_ZhOrNumOrLett(obj) {    //判断是否是汉字、字母、数字组成
425. **var** regu = "^[0-9a-zA-Z\u4e00-\u9fa5]+$";
426. **var** re = **new** RegExp(regu);
427. **if** (re.test(obj.value)) {
428. **return** **true**;
429. }
430. f\_alert(obj, "请输入汉字、字母或数字");
431. **return** **false**;
432. }
434. /\*
435. 用途：校验ip地址的格式
436. 输入：strIP：ip地址
437. 返回：如果通过验证返回true,否则返回false；
438. \*/
440. **function** f\_check\_IP(obj) {
441. **var** re = /^(\d+)\.(\d+)\.(\d+)\.(\d+)$/; //匹配IP地址的正则表达式
442. **if** (re.test(obj.value)) {
443. **if** (RegExp.$1 < 256 && RegExp.$2 < 256 && RegExp.$3 < 256 && RegExp.$4 < 256) **return** **true**;
444. }
445. f\_alert(obj, "请输入合法的计算机IP地址");
446. **return** **false**;
447. }

450. **function** f\_check\_IP(obj) {
451. **var** re = /^(\d+)\.(\d+)\.(\d+)\.(\d+)$/; //匹配IP地址的正则表达式
452. **if** (re.test(obj.value)) {
453. **if** (RegExp.$1 < 256 && RegExp.$2 < 256 && RegExp.$3 < 256 && RegExp.$4 < 256) **return** **true**;
454. }
455. f\_alert(obj, "请输入合法的计算机IP地址");
456. **return** **false**;
457. }
459. /\*
460. 用途：检查输入对象的值是否符合端口号格式
461. 输入：str 输入的字符串
462. 返回：如果通过验证返回true,否则返回false
463. \*/
465. **function** f\_check\_port(obj) {
466. **if** (!f\_check\_number(obj))
467. **return** **false**;
468. **if** (obj.value < 65536)
469. **return** **true**;
470. f\_alert(obj, "请输入合法的计算机IP地址端口号");
471. **return** **false**;
472. }

475. **function** f\_check\_port(obj) {
476. **if** (!f\_check\_number(obj))
477. **return** **false**;
478. **if** (obj.value < 65536)
479. **return** **true**;
480. f\_alert(obj, "请输入合法的计算机IP地址端口号");
481. **return** **false**;
482. }
484. /\*
485. 用途：检查输入对象的值是否符合网址格式
486. 输入：str 输入的字符串
487. 返回：如果通过验证返回true,否则返回false
488. \*/
490. **function** f\_check\_URL(obj) {
491. **var** myReg = /^((http:[/][/])?\w+([.]\w+|[/]\w\*)\*)?$/;
492. **if** (myReg.test(obj.value)) **return** **true**;
493. f\_alert(obj, "请输入合法的网页地址");
494. **return** **false**;
495. }

498. **function** f\_check\_URL(obj) {
499. **var** myReg = /^((http:[/][/])?\w+([.]\w+|[/]\w\*)\*)?$/;
500. **if** (myReg.test(obj.value)) **return** **true**;
501. f\_alert(obj, "请输入合法的网页地址");
502. **return** **false**;
503. }
505. /\*
506. 用途：检查输入对象的值是否符合E-Mail格式
507. 输入：str 输入的字符串
508. 返回：如果通过验证返回true,否则返回false
509. \*/
511. **function** f\_check\_email(obj) {
512. **var** myReg = /^([-\_A-Za-z0-9\.]+)@([\_A-Za-z0-9]+\.)+[A-Za-z0-9]{2,3}$/;
513. **if** (myReg.test(obj.value)) **return** **true**;
514. f\_alert(obj, "请输入合法的电子邮件地址");
515. **return** **false**;
516. }

519. **function** f\_check\_email(obj) {
520. **var** myReg = /^([-\_A-Za-z0-9\.]+)@([\_A-Za-z0-9]+\.)+[A-Za-z0-9]{2,3}$/;
521. **if** (myReg.test(obj.value)) **return** **true**;
522. f\_alert(obj, "请输入合法的电子邮件地址");
523. **return** **false**;
524. }
526. /\*
527. 要求：一、移动电话号码为11或12位，如果为12位,那么第一位为0
528. 二、11位移动电话号码的第一位和第二位为"13"
529. 三、12位移动电话号码的第二位和第三位为"13"
530. 用途：检查输入手机号码是否正确
531. 输入：
532. s：字符串
533. 返回：
534. 如果通过验证返回true,否则返回false
535. \*/
537. **function** f\_check\_mobile(obj) {
538. **var** regu = /(^[1][3][0-9]{9}$)|(^0[1][3][0-9]{9}$)/;
539. **var** re = **new** RegExp(regu);
540. **if** (re.test(obj.value)) {
541. **return** **true**;
542. }
543. f\_alert(obj, "请输入正确的手机号码");
544. **return** **false**;
545. }

548. **function** f\_check\_mobile(obj) {
549. **var** regu = /(^[1][3][0-9]{9}$)|(^0[1][3][0-9]{9}$)/;
550. **var** re = **new** RegExp(regu);
551. **if** (re.test(obj.value)) {
552. **return** **true**;
553. }
554. f\_alert(obj, "请输入正确的手机号码");
555. **return** **false**;
556. }
558. /\*
559. 要求：一、电话号码由数字、"("、")"和"-"构成
560. 二、电话号码为3到8位
561. 三、如果电话号码中包含有区号，那么区号为三位或四位
562. 四、区号用"("、")"或"-"和其他部分隔开
563. 用途：检查输入的电话号码格式是否正确
564. 输入：
565. strPhone：字符串
566. 返回：
567. 如果通过验证返回true,否则返回false
568. \*/
570. **function** f\_check\_phone(obj) {
571. **var** regu = /(^([0][1-9]{2,3}[-])?\d{3,8}(-\d{1,6})?$)|(^[0][1-9]{2,3}\d{3,8}(\d{1,6})?$)|(^\d{3,8}$)/;
572. **var** re = **new** RegExp(regu);
573. **if** (re.test(obj.value)) {
574. **return** **true**;
575. }
576. f\_alert(obj, "请输入正确的电话号码");
577. **return** **false**;
578. }

581. **function** f\_check\_phone(obj) {
582. **var** regu = /(^([0][1-9]{2,3}[-])?\d{3,8}(-\d{1,6})?$)|(^[0][1-9]{2,3}\d{3,8}(\d{1,6})?$)|(^\d{3,8}$)/;
583. **var** re = **new** RegExp(regu);
584. **if** (re.test(obj.value)) {
585. **return** **true**;
586. }
587. f\_alert(obj, "请输入正确的电话号码");
588. **return** **false**;
589. }
591. /\* 判断是否为邮政编码 \*/
593. **function** f\_check\_zipcode(obj) {
594. **if** (!f\_check\_number(obj))
595. **return** **false**;
596. **if** (obj.value.length != 6) {
597. f\_alert(obj, "邮政编码长度必须是6位");
598. **return** **false**;
599. }
600. **return** **true**;
601. }

604. **function** f\_check\_zipcode(obj) {
605. **if** (!f\_check\_number(obj))
606. **return** **false**;
607. **if** (obj.value.length != 6) {
608. f\_alert(obj, "邮政编码长度必须是6位");
609. **return** **false**;
610. }
611. **return** **true**;
612. }
614. /\*
615. 用户ID，可以为数字、字母、下划线的组合，
616. 第一个字符不能为数字,且总长度不能超过20。
617. \*/
619. **function** f\_check\_userID(obj) {
620. **var** userID = obj.value;
621. **if** (userID.length > 20) {
622. f\_alert(obj, "ID长度不能大于20");
623. **return** **false**;
624. }
626. **if** (!isNaN(userID.charAt(0))) {
627. f\_alert(obj, "ID第一个字符不能为数字");
628. **return** **false**;
629. }
630. **if** (!/^\w{1,20}$/.test(userID)) {
631. f\_alert(obj, "ID只能由数字、字母、下划线组合而成");
632. **return** **false**;
633. }
634. **return** **true**;
635. }

638. **function** f\_check\_userID(obj) {
639. **var** userID = obj.value;
640. **if** (userID.length > 20) {
641. f\_alert(obj, "ID长度不能大于20");
642. **return** **false**;
643. }
645. **if** (!isNaN(userID.charAt(0))) {
646. f\_alert(obj, "ID第一个字符不能为数字");
647. **return** **false**;
648. }
649. **if** (!/^\w{1,20}$/.test(userID)) {
650. f\_alert(obj, "ID只能由数字、字母、下划线组合而成");
651. **return** **false**;
652. }
653. **return** **true**;
654. }
656. /\*
657. 功能：验证身份证号码是否有效
658. 提示信息：未输入或输入身份证号不正确！
659. 使用：f\_check\_IDno(obj)
660. 返回：bool
661. \*/
663. **function** f\_check\_IDno(obj) {
664. **var** aCity = { 11: "北京", 12: "天津", 13: "河北", 14: "山西", 15: "内蒙古", 21: "辽宁", 22: "吉林", 23: "黑龙江", 31: "上海", 32: "江苏", 33: "浙江", 34: "安徽", 35: "福建", 36: "江西", 37: "山东", 41: "河南", 42: "湖北", 43: "湖南", 44: "广东", 45: "广西", 46: "海南", 50: "重庆", 51: "四川", 52: "贵州", 53: "云南", 54: "西藏", 61: "陕西", 62: "甘肃", 63: "青海", 64: "宁夏", 65: "新将", 71: "台湾", 81: "香港", 82: "澳门", 91: "国外" };
666. **var** iSum = 0;
667. **var** info = "";
668. **var** strIDno = obj.value;
669. **var** idCardLength = strIDno.length;
670. **if** (!/^\d{17}(\d|x)$/i.test(strIDno) && !/^\d{15}$/i.test(strIDno)) {
671. f\_alert(obj, "非法身份证号");
672. **return** **false**;
673. }
675. //在后面的运算中x相当于数字10,所以转换成a
676. strIDno = strIDno.replace(/x$/i, "a");
678. **if** (aCity[parseInt(strIDno.substr(0, 2))] == **null**) {
679. f\_alert(obj, "非法地区");
680. **return** **false**;
681. }
683. **if** (idCardLength == 18) {
684. sBirthday = strIDno.substr(6, 4) + "-" + Number(strIDno.substr(10, 2)) + "-" + Number(strIDno.substr(12, 2));
685. **var** d = **new** Date(sBirthday.replace(/-/g, "/"))
686. **if** (sBirthday != (d.getFullYear() + "-" + (d.getMonth() + 1) + "-" + d.getDate())) {
687. f\_alert(obj, "非法生日");
688. **return** **false**;
689. }
691. **for** (**var** i = 17; i >= 0; i--)
692. iSum += (Math.pow(2, i) % 11) \* parseInt(strIDno.charAt(17 - i), 11);
694. **if** (iSum % 11 != 1) {
695. f\_alert(obj, "非法身份证号");
696. **return** **false**;
697. }
698. }
699. **else** **if** (idCardLength == 15) {
700. sBirthday = "19" + strIDno.substr(6, 2) + "-" + Number(strIDno.substr(8, 2)) + "-" + Number(strIDno.substr(10, 2));
701. **var** d = **new** Date(sBirthday.replace(/-/g, "/"))
702. **var** dd = d.getFullYear().toString() + "-" + (d.getMonth() + 1) + "-" + d.getDate();
703. **if** (sBirthday != dd) {
704. f\_alert(obj, "非法生日");
705. **return** **false**;
706. }
707. }
708. **return** **true**;
709. }

712. **function** f\_check\_IDno(obj) {
713. **var** aCity = { 11: "北京", 12: "天津", 13: "河北", 14: "山西", 15: "内蒙古", 21: "辽宁", 22: "吉林", 23: "黑龙江", 31: "上海", 32: "江苏", 33: "浙江", 34: "安徽", 35: "福建", 36: "江西", 37: "山东", 41: "河南", 42: "湖北", 43: "湖南", 44: "广东", 45: "广西", 46: "海南", 50: "重庆", 51: "四川", 52: "贵州", 53: "云南", 54: "西藏", 61: "陕西", 62: "甘肃", 63: "青海", 64: "宁夏", 65: "新将", 71: "台湾", 81: "香港", 82: "澳门", 91: "国外" };
715. **var** iSum = 0;
716. **var** info = "";
717. **var** strIDno = obj.value;
718. **var** idCardLength = strIDno.length;
719. **if** (!/^\d{17}(\d|x)$/i.test(strIDno) && !/^\d{15}$/i.test(strIDno)) {
720. f\_alert(obj, "非法身份证号");
721. **return** **false**;
722. }
724. //在后面的运算中x相当于数字10,所以转换成a
725. strIDno = strIDno.replace(/x$/i, "a");
727. **if** (aCity[parseInt(strIDno.substr(0, 2))] == **null**) {
728. f\_alert(obj, "非法地区");
729. **return** **false**;
730. }
732. **if** (idCardLength == 18) {
733. sBirthday = strIDno.substr(6, 4) + "-" + Number(strIDno.substr(10, 2)) + "-" + Number(strIDno.substr(12, 2));
734. **var** d = **new** Date(sBirthday.replace(/-/g, "/"))
735. **if** (sBirthday != (d.getFullYear() + "-" + (d.getMonth() + 1) + "-" + d.getDate())) {
736. f\_alert(obj, "非法生日");
737. **return** **false**;
738. }
740. **for** (**var** i = 17; i >= 0; i--)
741. iSum += (Math.pow(2, i) % 11) \* parseInt(strIDno.charAt(17 - i), 11);
743. **if** (iSum % 11 != 1) {
744. f\_alert(obj, "非法身份证号");
745. **return** **false**;
746. }
747. }
748. **else** **if** (idCardLength == 15) {
749. sBirthday = "19" + strIDno.substr(6, 2) + "-" + Number(strIDno.substr(8, 2)) + "-" + Number(strIDno.substr(10, 2));
750. **var** d = **new** Date(sBirthday.replace(/-/g, "/"))
751. **var** dd = d.getFullYear().toString() + "-" + (d.getMonth() + 1) + "-" + d.getDate();
752. **if** (sBirthday != dd) {
753. f\_alert(obj, "非法生日");
754. **return** **false**;
755. }
756. }
757. **return** **true**;
758. }
760. /\*
761. \* 判断字符串是否符合指定的正则表达式
762. \*/
764. **function** f\_check\_formatStr(obj) {
765. **var** str = obj.value;
766. **var** dtype = obj.eos\_datatype;
767. **var** regu = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")"));    //指定的正则表达式
768. **var** re = **new** RegExp(regu);
769. **if** (re.test(str))
770. **return** **true**;
771. f\_alert(obj, "不符合指定的正则表达式要求");
772. **return** **false**;
773. }

776. **function** f\_check\_formatStr(obj) {
777. **var** str = obj.value;
778. **var** dtype = obj.eos\_datatype;
779. **var** regu = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")); //指定的正则表达式
780. **var** re = **new** RegExp(regu);
781. **if** (re.test(str))
782. **return** **true**;
783. f\_alert(obj, "不符合指定的正则表达式要求");
784. **return** **false**;
785. }
787. /\*
788. 功能：判断是否为日期(格式:yyyy年MM月dd日,yyyy-MM-dd,yyyy/MM/dd,yyyyMMdd)
789. 提示信息：未输入或输入的日期格式错误！
790. 使用：f\_check\_date(obj)
791. 返回：bool
792. \*/
794. **function** f\_check\_date(obj) {
795. **var** date = Trim(obj.value);
796. **var** dtype = obj.eos\_datatype;
797. **var** format = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")"));  //日期格式
798. **var** year, month, day, datePat, matchArray;
800. **if** (/^(y{4})(-|\/)(M{1,2})\2(d{1,2})$/.test(format))
801. datePat = /^(\d{4})(-|\/)(\d{1,2})\2(\d{1,2})$/;
802. **else** **if** (/^(y{4})(年)(M{1,2})(月)(d{1,2})(日)$/.test(format))
803. datePat = /^(\d{4})年(\d{1,2})月(\d{1,2})日$/;
804. **else** **if** (format == "yyyyMMdd")
805. datePat = /^(\d{4})(\d{2})(\d{2})$/;
806. **else** {
807. f\_alert(obj, "日期格式不对");
808. **return** **false**;
809. }
810. matchArray = date.match(datePat);
811. **if** (matchArray == **null**) {
812. f\_alert(obj, "日期长度不对,或日期中有非数字符号");
813. **return** **false**;
814. }
815. **if** (/^(y{4})(-|\/)(M{1,2})\2(d{1,2})$/.test(format)) {
816. year = matchArray[1];
817. month = matchArray[3];
818. day = matchArray[4];
819. } **else** {
820. year = matchArray[1];
821. month = matchArray[2];
822. day = matchArray[3];
823. }
824. **if** (month < 1 || month > 12) {
825. f\_alert(obj, "月份应该为1到12的整数");
826. **return** **false**;
827. }
828. **if** (day < 1 || day > 31) {
829. f\_alert(obj, "每个月的天数应该为1到31的整数");
830. **return** **false**;
831. }
832. **if** ((month == 4 || month == 6 || month == 9 || month == 11) && day == 31) {
833. f\_alert(obj, "该月不存在31号");
834. **return** **false**;
835. }
836. **if** (month == 2) {
837. **var** isleap = (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
838. **if** (day > 29) {
839. f\_alert(obj, "2月最多有29天");
840. **return** **false**;
841. }
842. **if** ((day == 29) && (!isleap)) {
843. f\_alert(obj, "闰年2月才有29天");
844. **return** **false**;
845. }
846. }
847. **return** **true**;
848. }

851. **function** f\_check\_date(obj) {
852. **var** date = Trim(obj.value);
853. **var** dtype = obj.eos\_datatype;
854. **var** format = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")); //日期格式
855. **var** year, month, day, datePat, matchArray;
857. **if** (/^(y{4})(-|\/)(M{1,2})\2(d{1,2})$/.test(format))
858. datePat = /^(\d{4})(-|\/)(\d{1,2})\2(\d{1,2})$/;
859. **else** **if** (/^(y{4})(年)(M{1,2})(月)(d{1,2})(日)$/.test(format))
860. datePat = /^(\d{4})年(\d{1,2})月(\d{1,2})日$/;
861. **else** **if** (format == "yyyyMMdd")
862. datePat = /^(\d{4})(\d{2})(\d{2})$/;
863. **else** {
864. f\_alert(obj, "日期格式不对");
865. **return** **false**;
866. }
867. matchArray = date.match(datePat);
868. **if** (matchArray == **null**) {
869. f\_alert(obj, "日期长度不对,或日期中有非数字符号");
870. **return** **false**;
871. }
872. **if** (/^(y{4})(-|\/)(M{1,2})\2(d{1,2})$/.test(format)) {
873. year = matchArray[1];
874. month = matchArray[3];
875. day = matchArray[4];
876. } **else** {
877. year = matchArray[1];
878. month = matchArray[2];
879. day = matchArray[3];
880. }
881. **if** (month < 1 || month > 12) {
882. f\_alert(obj, "月份应该为1到12的整数");
883. **return** **false**;
884. }
885. **if** (day < 1 || day > 31) {
886. f\_alert(obj, "每个月的天数应该为1到31的整数");
887. **return** **false**;
888. }
889. **if** ((month == 4 || month == 6 || month == 9 || month == 11) && day == 31) {
890. f\_alert(obj, "该月不存在31号");
891. **return** **false**;
892. }
893. **if** (month == 2) {
894. **var** isleap = (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
895. **if** (day > 29) {
896. f\_alert(obj, "2月最多有29天");
897. **return** **false**;
898. }
899. **if** ((day == 29) && (!isleap)) {
900. f\_alert(obj, "闰年2月才有29天");
901. **return** **false**;
902. }
903. }
904. **return** **true**;
905. }
907. /\*
908. 功能：校验的格式为yyyy年MM月dd日HH时mm分ss秒,yyyy-MM-dd HH:mm:ss,yyyy/MM/dd HH:mm:ss,yyyyMMddHHmmss
909. 提示信息：未输入或输入的时间格式错误
910. 使用：f\_check\_time(obj)
911. 返回：bool
912. \*/
914. **function** f\_check\_time(obj) {
915. **var** time = Trim(obj.value);
916. **var** dtype = obj.eos\_datatype;
917. **var** format = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")"));  //日期格式
918. **var** datePat, matchArray, year, month, day, hour, minute, second;
920. **if** (/^(y{4})(-|\/)(M{1,2})\2(d{1,2}) (HH:mm:ss)$/.test(format))
921. datePat = /^(\d{4})(-|\/)(\d{1,2})\2(\d{1,2}) (\d{1,2}):(\d{1,2}):(\d{1,2})$/;
922. **else** **if** (/^(y{4})(年)(M{1,2})(月)(d{1,2})(日)(HH时mm分ss秒)$/.test(format))
923. datePat = /^(\d{4})年(\d{1,2})月(\d{1,2})日(\d{1,2})时(\d{1,2})分(\d{1,2})秒$/;
924. **else** **if** (format == "yyyyMMddHHmmss")
925. datePat = /^(\d{4})(\d{2})(\d{2})(\d{2})(\d{2})(\d{2})$/;
926. **else** {
927. f\_alert(obj, "日期格式不对");
928. **return** **false**;
929. }
930. matchArray = time.match(datePat);
931. **if** (matchArray == **null**) {
932. f\_alert(obj, "日期长度不对,或日期中有非数字符号");
933. **return** **false**;
934. }
935. **if** (/^(y{4})(-|\/)(M{1,2})\2(d{1,2}) (HH:mm:ss)$/.test(format)) {
936. year = matchArray[1];
937. month = matchArray[3];
938. day = matchArray[4];
939. hour = matchArray[5];
940. minute = matchArray[6];
941. second = matchArray[7];
942. } **else** {
943. year = matchArray[1];
944. month = matchArray[2];
945. day = matchArray[3];
946. hour = matchArray[4];
947. minute = matchArray[5];
948. second = matchArray[6];
949. }
950. **if** (month < 1 || month > 12) {
951. f\_alert(obj, "月份应该为1到12的整数");
952. **return** **false**;
953. }
954. **if** (day < 1 || day > 31) {
955. f\_alert(obj, "每个月的天数应该为1到31的整数");
956. **return** **false**;
957. }
958. **if** ((month == 4 || month == 6 || month == 9 || month == 11) && day == 31) {
959. f\_alert(obj, "该月不存在31号");
960. **return** **false**;
961. }
962. **if** (month == 2) {
963. **var** isleap = (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
964. **if** (day > 29) {
965. f\_alert(obj, "2月最多有29天");
966. **return** **false**;
967. }
968. **if** ((day == 29) && (!isleap)) {
969. f\_alert(obj, "闰年2月才有29天");
970. **return** **false**;
971. }
972. }
973. **if** (hour < 0 || hour > 23) {
974. f\_alert(obj, "小时应该是0到23的整数");
975. **return** **false**;
976. }
977. **if** (minute < 0 || minute > 59) {
978. f\_alert(obj, "分应该是0到59的整数");
979. **return** **false**;
980. }
981. **if** (second < 0 || second > 59) {
982. f\_alert(obj, "秒应该是0到59的整数");
983. **return** **false**;
984. }
985. **return** **true**;
986. }

989. **function** f\_check\_time(obj) {
990. **var** time = Trim(obj.value);
991. **var** dtype = obj.eos\_datatype;
992. **var** format = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")); //日期格式
993. **var** datePat, matchArray, year, month, day, hour, minute, second;
995. **if** (/^(y{4})(-|\/)(M{1,2})\2(d{1,2}) (HH:mm:ss)$/.test(format))
996. datePat = /^(\d{4})(-|\/)(\d{1,2})\2(\d{1,2}) (\d{1,2}):(\d{1,2}):(\d{1,2})$/;
997. **else** **if** (/^(y{4})(年)(M{1,2})(月)(d{1,2})(日)(HH时mm分ss秒)$/.test(format))
998. datePat = /^(\d{4})年(\d{1,2})月(\d{1,2})日(\d{1,2})时(\d{1,2})分(\d{1,2})秒$/;
999. **else** **if** (format == "yyyyMMddHHmmss")
1000. datePat = /^(\d{4})(\d{2})(\d{2})(\d{2})(\d{2})(\d{2})$/;
1001. **else** {
1002. f\_alert(obj, "日期格式不对");
1003. **return** **false**;
1004. }
1005. matchArray = time.match(datePat);
1006. **if** (matchArray == **null**) {
1007. f\_alert(obj, "日期长度不对,或日期中有非数字符号");
1008. **return** **false**;
1009. }
1010. **if** (/^(y{4})(-|\/)(M{1,2})\2(d{1,2}) (HH:mm:ss)$/.test(format)) {
1011. year = matchArray[1];
1012. month = matchArray[3];
1013. day = matchArray[4];
1014. hour = matchArray[5];
1015. minute = matchArray[6];
1016. second = matchArray[7];
1017. } **else** {
1018. year = matchArray[1];
1019. month = matchArray[2];
1020. day = matchArray[3];
1021. hour = matchArray[4];
1022. minute = matchArray[5];
1023. second = matchArray[6];
1024. }
1025. **if** (month < 1 || month > 12) {
1026. f\_alert(obj, "月份应该为1到12的整数");
1027. **return** **false**;
1028. }
1029. **if** (day < 1 || day > 31) {
1030. f\_alert(obj, "每个月的天数应该为1到31的整数");
1031. **return** **false**;
1032. }
1033. **if** ((month == 4 || month == 6 || month == 9 || month == 11) && day == 31) {
1034. f\_alert(obj, "该月不存在31号");
1035. **return** **false**;
1036. }
1037. **if** (month == 2) {
1038. **var** isleap = (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
1039. **if** (day > 29) {
1040. f\_alert(obj, "2月最多有29天");
1041. **return** **false**;
1042. }
1043. **if** ((day == 29) && (!isleap)) {
1044. f\_alert(obj, "闰年2月才有29天");
1045. **return** **false**;
1046. }
1047. }
1048. **if** (hour < 0 || hour > 23) {
1049. f\_alert(obj, "小时应该是0到23的整数");
1050. **return** **false**;
1051. }
1052. **if** (minute < 0 || minute > 59) {
1053. f\_alert(obj, "分应该是0到59的整数");
1054. **return** **false**;
1055. }
1056. **if** (second < 0 || second > 59) {
1057. f\_alert(obj, "秒应该是0到59的整数");
1058. **return** **false**;
1059. }
1060. **return** **true**;
1061. }
1063. /\*判断当前对象是否可见\*/
1065. **function** isVisible(obj) {
1066. **var** visAtt, disAtt;
1067. **try** {
1068. disAtt = obj.style.display;
1069. visAtt = obj.style.visibility;
1070. } **catch** (e) { }
1071. **if** (disAtt == "none" || visAtt == "hidden")
1072. **return** **false**;
1073. **return** **true**;
1074. }

1077. **function** isVisible(obj) {
1078. **var** visAtt, disAtt;
1079. **try** {
1080. disAtt = obj.style.display;
1081. visAtt = obj.style.visibility;
1082. } **catch** (e) { }
1083. **if** (disAtt == "none" || visAtt == "hidden")
1084. **return** **false**;
1085. **return** **true**;
1086. }
1088. /\*判断当前对象及其父对象是否可见\*/
1090. **function** checkPrVis(obj) {
1091. **var** pr = obj.parentNode;
1092. **do** {
1093. **if** (pr == undefined || pr == "undefined") **return** **true**;
1094. **else** {
1095. **if** (!isVisible(pr)) **return** **false**;
1096. }
1097. } **while** (pr = pr.parentNode);
1098. **return** **true**;
1099. }

1102. **function** checkPrVis(obj) {
1103. **var** pr = obj.parentNode;
1104. **do** {
1105. **if** (pr == undefined || pr == "undefined") **return** **true**;
1106. **else** {
1107. **if** (!isVisible(pr)) **return** **false**;
1108. }
1109. } **while** (pr = pr.parentNode);
1110. **return** **true**;
1111. }
1113. /\* 弹出警告对话框，用户点确定后将光标置于出错文本框上， 并且将原来输入内容选中。\*/
1115. **function** f\_alert(obj, alertInfo) {
1116. **var** caption = obj.getAttribute("eos\_displayname");
1117. **if** (caption == **null**)
1118. caption = "";
1119. alert(caption + "：" + alertInfo + "！");
1120. obj.select();
1121. **if** (isVisible(obj) && checkPrVis(obj))
1122. obj.focus();
1123. }

1126. **function** f\_alert(obj, alertInfo) {
1127. **var** caption = obj.getAttribute("eos\_displayname");
1128. **if** (caption == **null**)
1129. caption = "";
1130. alert(caption + "：" + alertInfo + "！");
1131. obj.select();
1132. **if** (isVisible(obj) && checkPrVis(obj))
1133. obj.focus();
1134. }
1136. /\*\*
1137. \* 检测字符串是否为空
1138. \*/
1140. **function** isnull(str) {
1141. **var** i;
1142. **if** (str.length == 0)
1143. **return** **true**;
1144. **for** (i = 0; i < str.length; i++) {
1145. **if** (str.charAt(i) != ' ')
1146. **return** **false**;
1147. }
1148. **return** **true**;
1149. }

1152. **function** isnull(str) {
1153. **var** i;
1154. **if** (str.length == 0)
1155. **return** **true**;
1156. **for** (i = 0; i < str.length; i++) {
1157. **if** (str.charAt(i) != ' ')
1158. **return** **false**;
1159. }
1160. **return** **true**;
1161. }
1163. /\*\*
1164. \* 检测指定文本框输入是否合法。
1165. \* 如果用户输入的内容有错，则弹出提示对话框，
1166. \* 同时将焦点置于该文本框上，并且该文本框前面
1167. \* 会出现一个警告图标(输入正确后会自动去掉)。
1168. \*/
1170. **function** checkInput(object) {
1171. **var** image;
1172. **var** i;
1173. **var** length;
1175. **if** (object.eos\_maxsize + "" != "undefined") length = object.eos\_maxsize;
1176. **else** length = 0;
1178. **if** (object.eos\_isnull == "true" && isnull(object.value)) **return** **true**;
1180. /\* 长度校验 \*/
1181. **if** (length != 0 && strlen(object.value) > parseInt(length)) {
1182. f\_alert(object, "超出最大长度" + length);
1183. **return** **false**;
1184. }
1185. /\* 数据类型校验 \*/
1186. **else** {
1187. **if** (object.eos\_datatype + "" != "undefined") {
1189. **var** dtype = object.eos\_datatype;
1190. **var** objName = object.name;
1191. //如果类型名后面带有括号，则视括号前面的字符串为校验类型
1192. **if** (dtype.indexOf("(") != -1)
1193. dtype = dtype.substring(0, dtype.indexOf("("));
1194. //根据页面元素的校验类型进行校验
1195. **try** {
1196. **if** (eval("f\_check\_" + dtype + "(object)") != **true**)
1197. **return** **false**;
1198. } **catch** (e) { **return** **true**; }
1199. /\*  如果form中存在name前半部分相同，并且同时存在以"min"和"max"结尾的表单域，
1200. 那么视为按区间查询。即"min"结尾的表单域的值要小于等于"max"结尾的表单域的值。 \*/
1201. **if** (objName.substring((objName.length - 3), objName.length) == "min") {
1202. **var** objMaxName = objName.substring(0, (objName.length - 3)) + "max";
1203. **if** (document.getElementById(objMaxName) != undefined && document.getElementById(objMaxName) != "undefined") {
1204. **if** (checkIntervalObjs(object, document.getElementById(objMaxName)) != **true**)
1205. **return** **false**;
1206. }
1207. }
1208. }
1209. }
1210. **return** **true**;
1211. }

1214. **function** checkInput(object) {
1215. **var** image;
1216. **var** i;
1217. **var** length;
1219. **if** (object.eos\_maxsize + "" != "undefined") length = object.eos\_maxsize;
1220. **else** length = 0;
1222. **if** (object.eos\_isnull == "true" && isnull(object.value)) **return** **true**;
1224. /\* 长度校验 \*/
1225. **if** (length != 0 && strlen(object.value) > parseInt(length)) {
1226. f\_alert(object, "超出最大长度" + length);
1227. **return** **false**;
1228. }
1229. /\* 数据类型校验 \*/
1230. **else** {
1231. **if** (object.eos\_datatype + "" != "undefined") {
1233. **var** dtype = object.eos\_datatype;
1234. **var** objName = object.name;
1235. //如果类型名后面带有括号，则视括号前面的字符串为校验类型
1236. **if** (dtype.indexOf("(") != -1)
1237. dtype = dtype.substring(0, dtype.indexOf("("));
1238. //根据页面元素的校验类型进行校验
1239. **try** {
1240. **if** (eval("f\_check\_" + dtype + "(object)") != **true**)
1241. **return** **false**;
1242. } **catch** (e) { **return** **true**; }
1243. /\* 如果form中存在name前半部分相同，并且同时存在以"min"和"max"结尾的表单域，
1244. 那么视为按区间查询。即"min"结尾的表单域的值要小于等于"max"结尾的表单域的值。 \*/
1245. **if** (objName.substring((objName.length - 3), objName.length) == "min") {
1246. **var** objMaxName = objName.substring(0, (objName.length - 3)) + "max";
1247. **if** (document.getElementById(objMaxName) != undefined && document.getElementById(objMaxName) != "undefined") {
1248. **if** (checkIntervalObjs(object, document.getElementById(objMaxName)) != **true**)
1249. **return** **false**;
1250. }
1251. }
1252. }
1253. }
1254. **return** **true**;
1255. }
1257. /\* 检测表单中所有输入项的正确性，一般用于表单的onsubmit事件 \*/
1259. **function** checkForm(myform) {
1260. **var** i;
1261. **for** (i = 0; i < myform.elements.length; i++) {
1262. /\* 非自定义属性的元素不予理睬 \*/
1263. **if** (myform.elements[i].eos\_displayname + "" == "undefined") **continue**;
1264. /\* 非空校验 \*/
1265. **if** (myform.elements[i].eos\_isnull == "false" && isnull(myform.elements[i].value)) {
1266. f\_alert(myform.elements[i], "不能为空");
1267. **return** **false**;
1268. }
1269. /\* 数据类型校验 \*/
1270. **if** (checkInput(myform.elements[i]) == **false**)
1271. **return** **false**;
1272. }
1273. **return** **true**;
1274. }

1277. **function** checkForm(myform) {
1278. **var** i;
1279. **for** (i = 0; i < myform.elements.length; i++) {
1280. /\* 非自定义属性的元素不予理睬 \*/
1281. **if** (myform.elements[i].eos\_displayname + "" == "undefined") **continue**;
1282. /\* 非空校验 \*/
1283. **if** (myform.elements[i].eos\_isnull == "false" && isnull(myform.elements[i].value)) {
1284. f\_alert(myform.elements[i], "不能为空");
1285. **return** **false**;
1286. }
1287. /\* 数据类型校验 \*/
1288. **if** (checkInput(myform.elements[i]) == **false**)
1289. **return** **false**;
1290. }
1291. **return** **true**;
1292. }
1294. /\*\*
1295. \* 校验两个表单域数据的大小，目前只允许比较日期和数字。
1296. \* @param obj1 小值表单域
1297. \* @param obj2 大值表单域
1298. \*/
1300. **function** checkIntervalObjs(obj1, obj2) {
1301. **var** caption1 = obj1.getAttribute("eos\_displayname");
1302. **var** caption2 = obj2.getAttribute("eos\_displayname");
1303. **var** val1 = parseFloat(obj1.value);
1304. **var** val2 = parseFloat(obj2.value);
1305. // 非自定义属性的元素不予理睬
1306. **if** (obj1.eos\_displayname + "" == "undefined" || obj2.eos\_displayname + "" == "undefined") {
1307. **return** **false**;
1308. }
1309. // 日期类型的比较
1310. **if** (f\_check\_date(obj1) == **true** && f\_check\_date(obj2) == **true**) {
1311. **var** dtype = obj1.eos\_datatype;
1312. **var** format = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")"));  //日期格式
1313. val1 = getDateByFormat(obj1.value, format);
1314. dtype = obj2.eos\_datatype;
1315. format = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")"));  //日期格式
1316. val2 = getDateByFormat(obj2.value, format);
1317. **if** (val1 > val2) {
1318. obj2.select();
1319. **if** (isVisible(obj) && checkPrVis(obj))
1320. obj2.focus();
1321. alert(caption1 + "的起始日期不能大于其终止日期！");
1322. **return** **false**;
1323. }
1324. }
1325. // 数字类型的比较
1326. **if** ((isNaN(val1) && !isnull(val1)) || (isNaN(val2) && !isnull(val2))) {
1327. alert(caption1 + "的值不全为数字则不能比较！");
1328. **return** **false**;
1329. }
1330. **if** (val1 > val2) {
1331. obj2.select();
1332. **if** (isVisible(obj) && checkPrVis(obj))
1333. obj2.focus();
1334. alert(caption1 + "的起始值不能大于其终止值！");
1335. **return** **false**;
1336. }
1337. **return** **true**;
1338. }

1341. **function** checkIntervalObjs(obj1, obj2) {
1342. **var** caption1 = obj1.getAttribute("eos\_displayname");
1343. **var** caption2 = obj2.getAttribute("eos\_displayname");
1344. **var** val1 = parseFloat(obj1.value);
1345. **var** val2 = parseFloat(obj2.value);
1346. // 非自定义属性的元素不予理睬
1347. **if** (obj1.eos\_displayname + "" == "undefined" || obj2.eos\_displayname + "" == "undefined") {
1348. **return** **false**;
1349. }
1350. // 日期类型的比较
1351. **if** (f\_check\_date(obj1) == **true** && f\_check\_date(obj2) == **true**) {
1352. **var** dtype = obj1.eos\_datatype;
1353. **var** format = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")); //日期格式
1354. val1 = getDateByFormat(obj1.value, format);
1355. dtype = obj2.eos\_datatype;
1356. format = dtype.substring(dtype.indexOf("(") + 1, dtype.indexOf(")")); //日期格式
1357. val2 = getDateByFormat(obj2.value, format);
1358. **if** (val1 > val2) {
1359. obj2.select();
1360. **if** (isVisible(obj) && checkPrVis(obj))
1361. obj2.focus();
1362. alert(caption1 + "的起始日期不能大于其终止日期！");
1363. **return** **false**;
1364. }
1365. }
1366. // 数字类型的比较
1367. **if** ((isNaN(val1) && !isnull(val1)) || (isNaN(val2) && !isnull(val2))) {
1368. alert(caption1 + "的值不全为数字则不能比较！");
1369. **return** **false**;
1370. }
1371. **if** (val1 > val2) {
1372. obj2.select();
1373. **if** (isVisible(obj) && checkPrVis(obj))
1374. obj2.focus();
1375. alert(caption1 + "的起始值不能大于其终止值！");
1376. **return** **false**;
1377. }
1378. **return** **true**;
1379. }


1383. /\*根据日期格式，将字符串转换成Date对象。
1384. 格式：yyyy-年，MM-月，dd-日，HH-时，mm-分，ss-秒。
1385. （格式必须写全，例如:yy-M-d，是不允许的，否则返回null；格式与实际数据不符也返回null。）
1386. 默认格式：yyyy-MM-dd HH:mm:ss,yyyy-MM-dd。\*/
1388. **function** getDateByFormat(str) {
1389. **var** dateReg, format;
1390. **var** y, M, d, H, m, s, yi, Mi, di, Hi, mi, si;
1391. **if** ((arguments[1] + "") == "undefined") format = "yyyy-MM-dd HH:mm:ss";
1392. **else** format = arguments[1];
1393. yi = format.indexOf("yyyy");
1394. Mi = format.indexOf("MM");
1395. di = format.indexOf("dd");
1396. Hi = format.indexOf("HH");
1397. mi = format.indexOf("mm");
1398. si = format.indexOf("ss");
1399. **if** (yi == -1 || Mi == -1 || di == -1) **return** **null**;
1400. **else** {
1401. y = parseInt(str.substring(yi, yi + 4));
1402. M = parseInt(str.substring(Mi, Mi + 2));
1403. d = parseInt(str.substring(di, di + 2));
1404. }
1405. **if** (isNaN(y) || isNaN(M) || isNaN(d)) **return** **null**;
1406. **if** (Hi == -1 || mi == -1 || si == -1) **return** **new** Date(y, M - 1, d);
1407. **else** {
1408. H = str.substring(Hi, Hi + 4);
1409. m = str.substring(mi, mi + 2);
1410. s = str.substring(si, si + 2);
1411. }
1412. **if** (isNaN(parseInt(y)) || isNaN(parseInt(M)) || isNaN(parseInt(d))) **return** **new** Date(y, M - 1, d);
1413. **else** **return** **new** Date(y, M - 1, d, H, m, s);
1414. }

1417. **function** getDateByFormat(str) {
1418. **var** dateReg, format;
1419. **var** y, M, d, H, m, s, yi, Mi, di, Hi, mi, si;
1420. **if** ((arguments[1] + "") == "undefined") format = "yyyy-MM-dd HH:mm:ss";
1421. **else** format = arguments[1];
1422. yi = format.indexOf("yyyy");
1423. Mi = format.indexOf("MM");
1424. di = format.indexOf("dd");
1425. Hi = format.indexOf("HH");
1426. mi = format.indexOf("mm");
1427. si = format.indexOf("ss");
1428. **if** (yi == -1 || Mi == -1 || di == -1) **return** **null**;
1429. **else** {
1430. y = parseInt(str.substring(yi, yi + 4));
1431. M = parseInt(str.substring(Mi, Mi + 2));
1432. d = parseInt(str.substring(di, di + 2));
1433. }
1434. **if** (isNaN(y) || isNaN(M) || isNaN(d)) **return** **null**;
1435. **if** (Hi == -1 || mi == -1 || si == -1) **return** **new** Date(y, M - 1, d);
1436. **else** {
1437. H = str.substring(Hi, Hi + 4);
1438. m = str.substring(mi, mi + 2);
1439. s = str.substring(si, si + 2);
1440. }
1441. **if** (isNaN(parseInt(y)) || isNaN(parseInt(M)) || isNaN(parseInt(d))) **return** **new** Date(y, M - 1, d);
1442. **else** **return** **new** Date(y, M - 1, d, H, m, s);
1443. }
1445. /\*LTrim(string):去除左边的空格\*/
1447. **function** LTrim(str) {
1448. **var** whitespace = **new** String(" \t\n\r");
1449. **var** s = **new** String(str);
1451. **if** (whitespace.indexOf(s.charAt(0)) != -1) {
1452. **var** j = 0, i = s.length;
1453. **while** (j < i && whitespace.indexOf(s.charAt(j)) != -1) {
1454. j++;
1455. }
1456. s = s.substring(j, i);
1457. }
1458. **return** s;
1459. }

1462. **function** LTrim(str) {
1463. **var** whitespace = **new** String(" \t\n\r");
1464. **var** s = **new** String(str);
1466. **if** (whitespace.indexOf(s.charAt(0)) != -1) {
1467. **var** j = 0, i = s.length;
1468. **while** (j < i && whitespace.indexOf(s.charAt(j)) != -1) {
1469. j++;
1470. }
1471. s = s.substring(j, i);
1472. }
1473. **return** s;
1474. }
1476. /\*RTrim(string):去除右边的空格\*/
1477. **function** RTrim(str) {
1478. **var** whitespace = **new** String(" \t\n\r");
1479. **var** s = **new** String(str);
1481. **if** (whitespace.indexOf(s.charAt(s.length - 1)) != -1) {
1482. **var** i = s.length - 1;
1483. **while** (i >= 0 && whitespace.indexOf(s.charAt(i)) != -1) {
1484. i--;
1485. }
1486. s = s.substring(0, i + 1);
1487. }
1488. **return** s;
1489. }

1492. **function** RTrim(str) {
1493. **var** whitespace = **new** String(" \t\n\r");
1494. **var** s = **new** String(str);
1496. **if** (whitespace.indexOf(s.charAt(s.length - 1)) != -1) {
1497. **var** i = s.length - 1;
1498. **while** (i >= 0 && whitespace.indexOf(s.charAt(i)) != -1) {
1499. i--;
1500. }
1501. s = s.substring(0, i + 1);
1502. }
1503. **return** s;
1504. }
1506. /\*Trim(string):去除字符串两边的空格\*/
1507. **function** Trim(str) {
1508. **return** RTrim(LTrim(str));
1509. }

1512. **function** Trim(str) {
1513. **return** RTrim(LTrim(str));
1514. }