

# AXValidator

## new AXValidator()

□ [AXValidator.js, line 1](#)

axisj form validator

Version:

- v1.52.1

Author:

- tom@axisj.com, hwshin@superjump.co.kr

### Example

```
1. var myValidator = new AXValidator();
2. myValidator.setConfig({
3.     targetFormName : "myForm"    //{String} - validate를 수행하려는 form "name", ID가 아
    납니다.
4. });
```

## Extends

- [AXI](#)

## Methods

### static AXValidator.add()

□ [AXValidator.js, line 144](#)

validate 대상 아이템을 추가합니다.

Type	Description
<a href="#">JSObject</a>	example code <a href="#">참고</a>

### Example

```
1.
2. var jsObjectSample = {
3.     id: "userID",           //{string} - 아이템식별자
4.     label: "아이디",       //{string} - 아이템라벨
5.     config: {               //필요한 조합을 object로 정의합니다.
6.         required: true,     //[boolean=true] - 필수입력 체크
7.         number: true,       //[boolean=true] - 숫자입력 체크
8.         email: true,        //[boolean=true] - 이메일형식 체크
9.         hangul: true,       //[boolean=true] - 한글형식 체크
10.        engonly: true,       //[boolean=true] - 영문형식 체크
11.        residentno: true,    //[boolean=true] - 주민등록번호형식 체크
12.        foreignerno: true,   //[boolean=true] - 외국인번호형식 체크
13.        bizno: true,         //[boolean=true] - 사업자등록번호형식 체크
14.        phone: true,         //[boolean=true] - 전화번호형식 체크
```

```

15.         isdate: true,           //[boolean=true] - 날짜형식 체크
16.         zip: true,             //[boolean=true] - 우편번호형식 체크
17.         money: true,          //[boolean=true] - 숫자에 , 포함 체크
18.         earlierThan:{
19.             id: "targetId",      //{string} - 대상의 아이디. 현재 아이템의 값이 대상보다
    커야함
20.             label: "targetLabel" //{string} - 대상의 라벨
21.         },
22.         laterThan:{
23.             id: "targetId",      //{string} - 대상의 아이디. 현재 아이템의 값이 대상보다
    작아야함
24.             label: "targetLabel" //{string} - 대상의 라벨
25.         },
26.         min: true,             //[boolean=true] - 최소값
27.         max: true,             //[boolean=true] - 최대값
28.         minbyte: true,         //[boolean=true] - 최소바이트값
29.         maxbyte: true,         //[boolean=true] - 최대바이트값
30.         minlength: true,      //[boolean=true] - 최소길이
31.         maxlength: true       //[boolean=true] - 최대길이
32.     },
33.     realtime:{                 //[특정이벤트 발생시 액션정의
34.         event: "keydown",      //{String} - 발생하는 이벤트 종류
35.         response: function(){  //{fn} - 정의된 이벤트에 따른 실시간 이벤트 콜백함수
36.             //trace(this);
37.         }
38.     },
39.     onblur: function(){        //[fn] - 대상 아이템에 블러 이벤트 발생 콜백함수
40.         //trace(this);
41.     }
42. };
43.
44. var myValidator = new AXValidator();
45. myValidator.add({
46.     id: "userID",              //{string} - 아이템식별자
47.     label: "아이디",          //{string} - 아이템라벨
48.     config: {
49.         required: true,        //[boolean=true] - 필수입력 체크
50.         minbyte:10,            //[boolean=true] - 최소바이트값
51.         maxbyte:20             //[boolean=true] - 최대바이트값
52.     },
53.     realtime: {
54.         event: "keydown",      //{String} - 발생하는 이벤트 종류
55.         response: function () { //{fn} - 정의된 이벤트에 따른 실시간 이벤트 콜백함수
56.             if(this.result){
57.                 $("#userID_realtime").html("OK");
58.             }else{
59.                 $("#userID_realtime").html(this.message);
60.             }
61.             if (this.validateKey == "maxbyte" || this.validateKey == "maxlength") {
62.                 return false; //키 입력 중지
63.             } else {
64.                 return true; //키 입력 제어 안함
65.             }
66.         }
67.     },
68.     onblur: function(){        //[fn] - 대상 아이템에 블러 이벤트 발생 콜백함수

```

```

69.         trace(this);
70.     }
71. });
72.
73. myValidator.add({
74.     id:"enddate",    //{string} - 아이템 식별자
75.     label:"종료일",  //{string} - 아이템 라벨
76.     config:{
77.         isdate:true,    //[boolean=true] - 날짜형식 체크
78.         laterThan:{
79.             id:"regdate",    //{string} - 대상의 아이디. 현재 아이템의 값이 대상보다 작아야함
80.             label:"등록일"    //{string} - 대상의 라벨
81.         }
82.     },
83.     onblur: function(){    //[fn] - 대상 아이템에 블러 이벤트 발생 콜백함수
84.         trace(this);
85.     }
86. });

```

## static AXValidator.del()

□ [AXValidator.js, line 463](#)

validate 대상 아이템을 제거합니다.

Type	Description
JSObject	example code <a href="#">참고</a>

### Example

```
myValidator.del( {id:"userID"});    //{아이템식별자}
```

## static AXValidator.getErrorElement() {HTMLElement}

□ [AXValidator.js, line 972](#)

에러가 발생한 엘리먼트를 리턴합니다.

### Returns:

발생된 엘리먼트

### Example

```

var validateResult = myValidator.validate();
if (!validateResult) {
    var msg = myValidator.getErrorMessage();    //{에러메세지를 리턴합니다.}
    alert(msg);
    myValidator.getErrorElement().focus();    //{에러가 발생한 엘리먼트를 리턴합니다.}
    return false;
}else{
    alert( validateResult );
}

```

에러메세지를 리턴합니다.

#### Example

```
var validateResult = myValidator.validate();
if (!validateResult) {
    var msg = myValidator.getErrorMessage();    //에러메세지를 리턴합니다.
    alert(msg);
    myValidator.getErrorElement().focus();    //에러가 발생한 엘리먼트를 리턴합니다.
    return false;
}else{
    alert( validateResult );
}
```

validate 처리결과를 리턴합니다.

#### Returns:

규칙에 따라 (true|false) 로 결과를 리턴합니다.

#### Example

```
var validateResult = myValidator.validate();
```