MongoDB schema enforcement with Mongoose

	Common Data Types					
	Common to all	**String**	**Number**	**Date**	**Boolean**	**Array**
Attribute	*type*	lowercase	min	min		
	required	uppercase	max	max		
	default	trim	enum			
	validate	match				
	immutable	enum				
		minLength				
		maxLength				

```
email: {
    type: String,
    lowercase: true,
    match: [/^([\w-\.]+@([\w-]+\.)+[\w-]{2,4})?$/, '"email" must be a valid email format'],
    trim: true,
    required: [true, '"primary_email" is required'],
},
color: {
    type: String,
    enum: {
        values: ['red', green', 'blue'],
        message: '{VALUE} is not a valid color value',
    },
    required: true,
    immutable: true,
},
credit_score: {
    type: Number,
    min: 300,
    max: 850,
    default: 650,
},
member_since: {
    type: Date,
    immutable: true,
    default: Date.now,
},
active: {
    type: Boolean,
    default: true,
}
```

'match': is special validator type that takes a regex

Each attribute can be given custom message by changing value into an array with [value, message]

'type': is special attribute that uses keywords in mongoose.Types and user-defined Schemas

'immutable': value can only be set for new documents

```
function isAplhaNumeric (value) {
    return (value.match(/^[a-zA-Z0-9]+$/));
}

function hasKeyword (value) {
    return (value.includes('keyword'));
}

const usernameValidators = [
  { validator: isAlphaNumeric, msg: '"username" can only contain letters...' },
  { validator: hasKeyword, msg: 'invalid keyword in "username"' }
]

username: {
    type: String,
    validate: usernameValidators,
    trim: true,
},

challenge_words: {
    type: [String],
    default: undefined,
},
```

Functions return boolean value

Complex data validation can be accomplished by an array of {validator: ... , msg: ...}

Arrays are defined by [ ] syntax with a type that can be mongoose.Types or user-defined Schemas

Arrays have a default value of empty [ ], to override assign as undefined