

Object Validation in JavaScript

Xavier Dutreilh

Motivation

Check the validity
of input data

Ensure the schema
of input data

Subset the values
of input data

State of the art

Ad-hoc validations

Framework validations

Platform validations

Browser validations

Schema validations

Introduction to joi

Lightweight module
for schema validation

Only four production
dependencies

Initially, part of **hapi**,
became a module later

Actively maintained with
more than 140 releases

Actively used with more
than 1.7M downloads
per month on [npm](#)

Examples

Definition

```
const joi = require('joi');
```

```
const userSchema = joi.object().keys({
  username: joi.string().alphanum().required(),
  email: joi.string().email().required(),
  password: joi.string().regex(/^w{16,}$/).required(),
  role: joi.string().valid('admin', 'manager', 'user').required(),
  birthdate: joi.string().regex(/^d{4}-d{2}-d{2}$/),
});
```

Usage

```
const user = {  
  username: 'xavierdutreilh',  
  email: 'xavier@dutreilh.com',  
  password: 'abc123ABC123abcZ',  
  role: 'admin',  
  birthdate: '1984-07-27',  
};
```

```
const validation = joi.validate(user, userSchema);  
// const validation = joi.assert(user, userSchema);  
// const validation = joi.attempt(user, userSchema);
```



```
{
  error: null,
  value: {
    username: 'xavierdutreilh',
    email: 'xavier@dutreilh.com',
    password: 'abc123ABC123abcZ',
    role: 'admin',
    birthdate: '1984-07-27',
  },
}
```

```
const user2 = {  
  email: 'xavier@dutreilh.com',  
  password: 'abc123ABC123abcZ',  
  role: 'admin',  
  birthdate: '1984-07-27',  
};
```

```
const validation = joi.validate(user2, userSchema);  
// const validation = joi.assert(user2, userSchema);  
// const validation = joi.attempt(user2, userSchema);
```

```
{
  error: {
    isJoi: true,
    name: 'ValidationError',
    details: [{
      message: '"username" is required', path: 'username',
      type: 'any.required', context: {key: 'username'},
    }],
    _object: user2,
    annotate: function() {...}, // annotate errors in _object
  },
  ...
}
```

```
const validation = joi.validate(user, userSchema, {  
  // return all errors (by default, stop on first error)  
  abortEarly: false,  
  // allow unknown elements (by default, disallow them)  
  allowUnknown: true,  
  // remove unknown elements (by default, keep them)  
  stripUnknown: true,  
  // set default presence requirement (by default, optional)  
  presence: 'required', // optional, required, or forbidden  
});
```

Specialization

```
const optionalSchema = userSchema.optionalKeys('role');
```

Concatenation


```
const customerSchema = userSchema.concat(  
  joi.object().keys({  
    givenName: joi.string().default('John').required(),  
    familyName: joi.string().default('Doe').required(),  
  })  
);
```

Composition

```
const groupSchema = joi.object().keys({  
  name: joi.string().required(),  
  description: joi.string(),  
});
```

```
const userSchema = joi.object().keys({  
  ...  
  groups: joi.array().items(groupSchema).required(),  
});
```

Extension

```
const owasp = joi.extend({
  base: joi.string(),
  name: 'password',
  language: {strong: 'needs to be a strong password'},
  rules: [{name: 'strong', validate: validate}],
});
```

```
const password = require('owasp-password-strength-test');  
  
function validate(params, value, state, options) {  
  if (password.test(value).strong) return value;  
  
  return this.createError(  
    'password.strong', {v: value}, state, options  
  );  
}
```

```
const userSchema = joi.object().keys({  
  ...  
  password: owasp.password().strong().required(),  
  ...  
});
```

Documentation

See the [API Reference](#)

Integration

Express

```
const joi = require('joi');
```

```
app.post('/users', (request, response, next) => {  
  const validation = joi.validate(request.body, userSchema);  
  
  if (validation.error) return next(validation.error);  
  
  User.create(validation.value).then((user) => {  
    response.json(user);  
  });  
});
```

AngularJS

```
const joi = require('joi'); // loader required (webpack)  
// alternative: joi-browser but bundled files are outdated
```

```
class UserFormController {  
  ...  
  save() {  
    const validation = joi.validate(this.user, userSchema);  
  
    if (validation.error) {  
      this.error = this.humanizeError(validation.error);  
    } else {  
      this.UserService.save(validation.value);  
    }  
  }  
}
```



```
class UserFormController {
```

```
...
```

```
humanizeError(error) {
```

```
  return error.details.reduce((obj, detail) => {
```

```
    obj[detail.path] = this.getText.getString(detail.message);
```

```
    return obj;
```

```
  }, {});
```

```
}
```

```
...
```

```
}
```

Conclusion

Always validate input data
before manipulation

Implement data validation
in your forms

Use schema validation
when both front-end and
back-end require them

Use schema validation
when input data comes
from various sources

Thank you!

Xavier Dutreilh

xavier@dutreilh.com

<http://xavier.dutreilh.com/>