Cheatography

# SailsJS Cheat Sheet

by ProLoser via cheatography.com/1600/cs/3124/

## Model Validations

**after:** *{Date object}*

**alpha:** *{boolean}*

**alphadashed:** *{boolean}*

**alphanumeric:** *{boolean}*

**alphanumericdashed:** *{boolean}*

**array:** *{boolean}*

> strings formatted as arrays don't pass

**before:** *{Date object}*

**boolean:** *{boolean}*

> strings fail

**contains:** *{string?RegEx?}*

**creditcard:** *{boolean}*

**date:** *{boolean}*

**datetime:** *{boolean}*

**decimal:** *{boolean}*

**email:** *{boolean}*

**empty:** *{boolean}*

**equals:** *{mixed}*

> === comparison

**falsey:** *{boolean}*

**finite:** *{boolean}*

> can be coerced to a finite number

**float:** *{boolean}*

**hexadecimal:** *{boolean}*

**hexColor:** *{boolean}*

**in:** *{array}*

**int:** *{boolean}*

> alias for integer

**integer:** *{boolean}*

## Model Validations (cont)

**ip:** *{boolean}*

> IPv4 or IPv6

**ipv4:** *{boolean}*

**ipv6:** *{boolean}*

**is:** *{RegEx}*

**json:** *{boolean}*

**len:** *{?}*

> is integer > param1 && < param2.
> **(Where are params defined?)**

**lowercase:** *{boolean}*

**max:** *{number}*

**minLength:** *{int}*

**not:** *{RegEx}*

**notContains:** *{string?RegEx?}*

**notEmpty:** *{boolean}*

**notIn:** *{string|array}*

**notNull:** *{boolean}*

**notRegex:** *{boolean?}*

**null:** *{boolean}*

**number:** *{boolean}*

> NaN is considered a number

**numeric:** *{boolean}*

> string contains only numbers

**object:** *{boolean}*

**regex:** *{boolean?}*

**protected:** *{boolean}*

> remove attribute when toJSON is called on instance

**required:** *{boolean}*

> during creation only

**string:** *{boolean}*

## Model Validations (cont)

**text:** *{boolean}*

**truthy:** *{boolean}*

**undefined:** *{boolean}*

**unique:** *{boolean}*

**uppercase:** *{boolean}*

**url:** *{boolean}*

**urlish:** *{boolean}*

> /^\s([^\/]+.)+.+\s*$/g

**uuid:** *{boolean}*

> UUID v3, v4 or v5

**uuidv3:** *{boolean}*

**uuidv4:** *{boolean}*

Define custom validation using the `types` property of the model

## Websocket PupSub Methods

**message**(record|id,message,*[request]*)

> Message: {id,verb:'messaged',data}

**publishCreate**(data,*[request]*)

> Message: {id,verb:'created',data}

**publishUpdate**(id,*[data]*,*[request]*,*[options]*)

> Message: {id,verb:'updated',data:object,previous:object}

**publishAdd**(id,attributes,idAdded,*[request]*,*[options]*)

> Message: {id,verb:'addedTo',attribute,addedId}

**publishDestroy**(id,*[request]*,*[options]*)

> Message: {id,verb:'destroyed',previous}

**publishRemove**(id,attribute,removedId,*[request]*,*[options]*)

> Message: {id,verb:'removedFrom',attribute,removedId}

By **ProLoser**
cheatography.com/proloser/

Published 2nd January, 2015.
Last updated 12th May, 2016.

Cheatography

# SailsJS Cheat Sheet

by ProLoser via cheatography.com/1600/cs/3124/

## Websocket PupSub Methods (cont)

**subscribe**(request,record(s),*[context(s)]*)

Subscribes to record(s) changes. *Works for socket requests only.*

**unsubscribe**(request,record(s),*[context(s)]*)

Unsubscribes from record(s) changes. *Works for socket requests only.*

**subscribers**(record|id,*[context(s)]*) *{Array-(Socket)}*

**watch**(request)

Subscribe to **publishCreate()**. Works for socket requests only.

**unwatch**(request)

Unsubscribe to **publishCreate()**. Works for socket requests only.

http://sailsjs.org/#/documentation/reference/websockets/resourceful-pubsub

## Model Lifecycle Callbacks

**beforeValidate:** *{fn(values, next)}*

`create` and `update`

**afterValidate:** *{fn(values, next)}*

`create` and `update`

**beforeCreate:** *{fn(values, next)}*

**afterCreate:** *{fn(newlyInsertedRecord, next)}*

**beforeUpdate:** *{fn(valuesToUpdate, next)}*

**afterUpdate:** *{fn(updatedRecord, next)}*

**beforeDestroy:** *{fn(criteria, next)}*

**afterDestroy:** *{fn(destroyedRecords, next)}*

## Waterline Model Methods

**query**(query, callback(err, results))

Only works with PostgreSQL and mySQL

**native**(callback(err, collection))

Only works with MongoDB. For low-level usage do `requir e(' mon godb')` directly.

**count**([criteria], [callback]) *{query}*

**find**(criteria, [callback]) *{query}*

**findOne**(criteria, [callback]) *{query}*

**findOrCreate**(criteria, [callback]) *{query}*

**create**(data, [callback]) *{query}*

**destroy**(criteria, [callback]) *{query}*

**update**(criteria, data, [callback]) *{query}*

**stream**(criteria, [options]) *{stream}*

If a callback **is not** passed, most methods return chainable `query` object which ends with `.exec( cal lback)`

## Waterline Record Methods

**add**(*{object|id}*)

Add a many2many relationship

**remove**(*{object|id}*)

Remove a many2many relationship

**toJSON**() *{clonedObject}*

Contains instance methods

**toObject**() *{clonedData}*

Does not contain instance methods

**validate**(callback(err))

**save**(callback)

persists any changes to the database. Required for **add()** / **remove()**

## Waterline Queries

**exec**(callback)

Executes a query at the end of a query chain.

**limit**(int) *{query}*

**populate**(foreignKey, *[query]*) *{query}*

**populateAll**(*[query]*) *{query}*

**skip**(int) *{query}*

**sort**(sortString) *{query}*

**where**(criteria) *{query}*

**then**(callback) *{query}*

Bluebird Promise Method

**catch**(callback) *{query}*

Bluebird Promise Method

Query objects are returned by waterline methods **if a callback was not passed**.

## Model Settings

**migrate:** *{string}*

**safe:** never auto-migrate db. **alter:** auto-migrate db and *attempt* to keep existing. **drop:** drop all data on every lift.

**schema:** *{boolean}*

only specified attributes are saved

**connection:** *{string}*

use specified connection config

**identity:** *{string}*

lowercase version of filename

**globalId** *{string}*

global variable name if enabled

**autoPK:** *{boolean}*

**autoCreatedAt:** *{boolean}*

**autoUpdatedAt:** *{boolean}*

By **ProLoser**
cheatography.com/proloser/

Published 2nd January, 2015.
Last updated 12th May, 2016.

Cheatography

# SailsJS Cheat Sheet

by ProLoser via cheatography.com/1600/cs/3124/

## Model Settings (cont)

**tableName:** *{string}*

table or collection to be used

**attributes:** *{object}*

http://sailsjs.org/#/documentation/concepts/ORM/model-settings.html

## Model Attributes

**type:** *{string}*

string

text

integer

float

date

datetime

boolean

boolean

binary

array

json

email

**defaultsTo:** *{mixed}*

**autoIncrement:** *{boolean}*

requires `type: 'integer'`

**unique:** *{boolean}*

**primaryKey:** *{boolean}*

requires autoPK set to false

**enum:** *{array[string]}*

**size:** *{integer}*

**columnName:** *{string}*

Each attribute can have an object definition or just a string specifying containing type

## Query Language

**Where:**

Mode.find({ where: conditions })

{ property: 'value' }

**{ property: { '*comparison*': 'value' } }**

Complex comparison options. Works for dates: `{ '>': new Date(' 2/4 /2014') }`

{ '<' | 'lessThan' : value }

{ '<=' | 'lessThanOrEqual' : value }

{ '>' | 'greaterThan' : value }

{ '>=' | 'greaterThanOrEqual' : value }

{ '!' | 'not' : value }

{ 'like|contains|startsWith|endsWith': value }

**like:** `'sea%rch'` **contains:** `'%search%'` **startsWith:** `'search%'` **endsWith:** `'%search'`

{ property: ['value1', 'value2'] }

Value in array

{ property: { '!' : ['value1', 'value2'] } }

Value not in array

{ or: [ { prop1: 'val1' }, { prop2: val2 } ] }

**Limit:**

Mode.find({ limit: integer })

**Skip:**

Mode.find({ skip: integer })

**Sort:**

Mode.find({ sort: sortString })

{ sort: 'property' | 'property ASC' }

{ sort: 'property DESC' }

{ sort: { 'property1' : 1, 'property2': 0 } }

## Model Associations

**Has-One:**

**model:** 'model'

**Has-Many:**

**collection:** 'model'

**via:** 'foreignPropertyName'

**[*dominant*]:** true

Optionally used if both sides use `via` property

Associations are defined in the attributes property of a model

By **ProLoser**
cheatography.com/proloser/

Published 2nd January, 2015.
Last updated 12th May, 2016.