

### Response Example

```
let res = {
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-513-21311-3",
        "price": 8.99
      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ]
  }
}
```

### Response Example (cont)

```
> ],
  "bicycle": {
    "color": "red",
    "price": 19.95
  }
};
```

### Extract data

```
let firstAuthor = res.store.book[0].author; //
Nigel Rees
let lastAuthor = res.store.book[res.store.book.length - 1].author; // J. R. R. Tolkien
let firstBook = res.store.book[0];
/*{
  category: 'reference',
  author: 'Nigel Rees',
  title: 'Sayings of the Century',
  price: 8.95
}
*/
let books = res.store.book;
let titleList = [];
books.forEach((element) => {
  if (element.price < 15 && element.category === "fiction") {
    titleList.push(element.title);
  }
});
// [ 'Sword of Honour', 'Moby Dick' ]
let bookList = _.filter(books, (element) => {
  return element.price < 15 && element.category === "fiction";
});
```

### Extract data (cont)

```
> let titles = _.map(bookList, _property("title"));
//or
let titles = _.map(bookList, "title");
console.log(titles);
//[ 'Sword of Honour', 'Moby Dick' ]
let isbn = [];
books.forEach((element) => {
  if (element.isbn !== undefined) {
    isbn.push(element.isbn);
  }
});
//[ '0-553-21311-3', '0-395-19395-8' ]
```

### Working with time

```
var moment = require('moment');
now = moment();

// 07/05/2021
console.log(now.format("D D/M M/Y YY"));
// tăng 1 ngày và 1 giờ so với thời điểm hiện tại
next_1_day_1hour = now.add(1, "d").add(1, "h");
console.log(next_1_day_1hour.format("D - D/M M/YYYY HH:mm:ss"));
// giảm 1 ngày so với thời điểm hiện tại
yesterday = now.subtract(1, "days");
console.log(yesterday.format("D D/M M/Y - YY"));
//mili second
console.log(moment().valueOf());
//second
console.log(moment().unix());
```

### Branching

```
// Call request A
postman.sendRequest("A ");
// Stop
postman.sendRequest(null);
```

### Assertion

```
pm.test("name of the test", () => {
  //code to make assertion
});
//skip test
pm.test.skip("name of the test", () => {
  //code to make assertion
});
```

### Get request information

```
request.name
//request A
request.url
//http s://postman-echo.com/post
request.method
//POST
request.headers
/*
{
  content-type: "application/json",
  user-agent: "PostmanRuntime/7.28.4",
  accept: "/",
  host: "postman-echo.com",
  ...
}
```



### Get request information (cont)

```
> */
request.headers['content-type']
//application/json
request.headers.cookie
//sails.sid=s%3A__llvuQ-_d3y8a2MCRW_ZmNWay5zb_OC.sMfNnx
//Get body in json
pm.request.body.raw
/*
{
  phone: "0931831823",
  code: "402839ouroiewr"
}
*/
//Get body in urlencoded
pm.request.body.urlencoded
//a=1&b=2
//Get body in form-data
pm.request.body.formData.get("name")
```

### Chaijs

`expect(object1).keyword(object2)`

**Chain Keyword:** to, be, been, is, that, which, and, has, have, with, at, of, same, but, does, still, also

**Support Keyword:** `.not`, `.deep`, `.nested`, `.own`, `.ordered`, `.any`, `.all`

**Action Keyword:** `.a(type[, msg])` `.include(val[, msg])` `.ok` `.true` `.false` `.null` `.undefined` `.nan` `.exist` `.empty` `.equal(val[, msg])` `.eql(obj[, msg])` `.above(n[, msg])` `.least(n[, msg])` `.below(n[, msg])` `.most(n[, msg])` .....

### Example

```
pm.expect(pm.response.code).eql(200);
pm.expect(res.status).eql(200);
let firstAuthor = res.store.books[0].author;
pm.expect(firstAuthor).toEqual("Nigel Rees");
pm.expect(res.status).above(3);
let testQuantity = res.status;
pm.expect(testQuantity).toBeTruthy();
const expect = require("chai").expect;
let obj = {
  category: "reference",
  author: "Nigel Rees",
  title: "Sayings of the Century",
  price: 8.95
}
pm.test("test first book", () =>{
  let firstBook = res.store.books[0];
  expect(firstBook).eql(obj);
});
let books = res.store.books;
let titles = _.map(books, _.property("title"));
/*
[
  "Sayings of the Century",
  "Sword of Honour",
  "Moby Dick",
  "The Lord of the Rings"
]
*/
expect(titles).include("The Lord of the Rings");
```



### Example (cont)

```
> titles.forEach((e) => expect(e).eql("The Lord of the Rings"));
titles.forEach((e) => expect(e).string("o"));
```

### Check schema

```
//response
{
  " id": 1,
  " fir stN ame ": " Ver non ",
  " las tNa me": " Har per ",
  " ema il": " ege sta s.r hon cus.Pr oin @ma -
ssa Qui squ epo rtt ito r.o rg",
  " pro gra mme ": " Fin ancial Analys is",
  " cou rse s": [
    " Acc oun tin g",
    " Sta tis tic s"
  ]
}
//Test
const schema = {
  " typ e": " obj ect ",
  " pro per tie s": {
    " id": {"ty pe": " num ber "},
    " fir stN ame ": {"ty pe": " str ing "},
    " las tNa me": {"ty pe": " str ing "},
    " ema il": {"ty pe": " str ing "},
    " pro gra mme ": {"ty pe": " str ing "},
    " cou rse s": {"ty pe": " arr ay"}
  },
  // Field nào là bắt buộc thì liệt kê ở phần require
  " req uir e": ["id ", " cou rse s"]
};
```

### Check schema (cont)

```
> pm.test("Validate schema", () => {
  pm.response.to.have.jsonSchema(schema);
});
```

### Newman

```
const newman = require("newman");
const moment = require("moment-timezone");
let date_time = moment().tz("Asia/Hong_Kong -
Min h").format("DD-MM-YY YY_HH-mm-ss");
let fileName = `report-c las sl- ${date_time}`;
newman.run(
  {
    col lec tion: require("./ col -
lle cti ons /cl ass 1.p ost man _col le cti on.j -
so n"),
    env iro nment: require("./ en -
v/D eve lop men t.p ost man _en vir onm ent.js -
on"),
    rep orters: ["cli", " htm lex -
tra "],
    rep orter: {
      htm lextra: {
        export: `./tes -
t/r epo rt/ ${f ile Nam e}.h tml`
      }
    },
    (err, summary) => {
      if (err) throw err;
      console.log(summary.run.f -
ai lures);
    }
  }
);
```



### Get response information

```

pm.response.code
//200
pm.res pon se.s tatus
//OK
pm.res pon se.r es pon seTime
//295
pm.res pon se.r es pon seSize
//633
//Get body in string
pm.res pon se.t ext()
//or
respon seBody
// {"ar gs": {}, " dat a":{ " pho ne": " 093 -
183 " ,"co de": " 402 8ew r"}} "
//Get body in json
pm.res pon se.j son()
//or
JSON.p ars e(r esp ons eBody)
/*
{
  " arg s": {},
  " dat a": {
    " pho ne": " 093 183 182 3",
    " cod e": " 402 839 our oie wr"
  }
}
*/
//Get response headers
respon seH eaders
/*

```

### Get response information (cont)

```

> {
  "Date": "Wed, 13 Jul 2022 02:00:26 GMT",
  "Content-Type": "application/json; charset=utf-8",
  "Content-Length": "588",
  "Connection": "keep-alive",
  "ETag": "W/\"24c-5jASVJWFOfv/weD+EO/h0FSWRLk\"",
  "Vary": "Accept-Encoding",
  "set-cookie": "sails.sid=s%3ATHifsul4oQ2kju1Pf2KjQ; Path=/;
HttpOnly"
}
*/

```

