**WINSTON :**

1. **Logging :**

We can do logging by default Winston logger.

Also we can create our own custom logger where we can modify the content shown according to our needs like: time stamp, profile, filters, process id etc,

A **transport** is essentially a storage device for your logs. Each instance of a **Winston logger** can have multiple transports configured at different levels. For example, one may want error logs to be stored in a persistent remote location (like a database), but all logs output to the console or a local file.

1. **Meta Data:**

Winston can additionally log JSON metadata objects.

winston.log('info', 'Test Log Message', { anything: 'This is metadata' });

1. **Multiple transports of the same type**

It is possible to use multiple transports of the same type

e.g. winston.transports.File by passing in a custom name when you construct the transport.

var logger = new (winston.Logger)({

transports: [

new (winston.transports.File)({

name: 'info-file',

filename: 'filelog-info.log',

level: 'info'

}),

new (winston.transports.File)({

name: 'error-file',

filename: 'filelog-error.log',

level: 'error'

})

]

});

If you later want to remove one of these transports you can do so by using the string name. e.g.:

logger.remove('info-file');

1. **Profiling**

Winston also provide profiling now all the log have test profile,by default winstion have default profile “INFO ”.

winston.profile('test');

setTimeout(function () {

//

// Stop profile of 'test'. Logging will now take place:

// "17 Jan 21:00:00 - info: test duration=1000ms"

//

winston.profile('test');

}, 1000);

1. **String interpolation**

We can format our string in whatever format we want and we the o/p acc. To our requirement.

logger.log('info', 'test message', 'first', 'second', {number: 123});

// info: test message first second

// meta = {number: 123}

logger.log('info', 'test message %s, %s', 'first', 'second', {number: 123}, function(){});

// info: test message first, second

// meta = {numer: 123}

// callback = function(){}

1. Querying logs is easy
2. var options = {
3. from: new Date - 24 \* 60 \* 60 \* 1000,
4. until: new Date,
5. limit: 10,
6. start: 0,
7. order: 'desc',
8. fields: ['message']
9. };
10. //
11. // Find items logged between today and yesterday.
12. //
13. winston.query(options, function (err, results) {
14. if (err) {
15. throw err;
16. }
17. console.log(results);
18. });

7. Streaming logs

8. Handling Uncaught Exception with Winston (system exception etc.. )

9. **multiple Loggers in Winston (set the multiple logger in file and get the output )**

Banyan :

1. Not very human friendly logs.

{"name":"myapp","hostname":"pwony-2","pid":12616,"level":30,"msg":"hi","time":"2014-05-26T17:58:32.835Z","v":0}

U can pipe the stdin output with banyan and get the readable output.

Main benefit is we do not need to reconfigure it again and again for different environment

1. Child logger : for maintain the separate log for different part of application.
2. Streaming easy same a Winston.