

# TESTING CHEAT SHEET

Framework for convenient unit testing. This cheat sheet summarizes commonly used utility Testing command line instructions and test suite structure for quick reference.

## INSTALLATION OF UTILITY TESTING

To install utility Testing an installed NodeJS and NPM are needed.

<b>\$ npm install -g wTesting</b>
Global installation of utility Testing by NPM.

## MAIN COMMANDS

All commands of the utility start with **tst** .

<b>\$ tst .help</b>
Get help.
<b>\$ tst .help [ command ]</b>
Get help on a specific command.
<b>\$ tst .suites.list [ path ]</b>
Find test suites at a specific path.
<b>\$ tst .run [ path ]</b>
Run test suites found at a specific path.
<b>\$ tst .imply [ options... ] .run [ path ]</b>
Change state or imply variable value.

## RUNNING TESTS

<b>\$ tst .run [ path to a test file ( directory )    path with glob ]</b>
<b>\$ node [ path to a test file ]</b>
Running of test suite ( test suites ) by utility Testing and NodeJS.

## PRIMARY RUNNING OPTIONS

To control testing the running options is used.

<b>\$ tst .imply verbosity:[ number ] .run [ path ]</b>
<b>\$ tst .imply v:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] verbosity:[ number ]</b>
<b>\$ tst .run [ path ] v:[ number ]</b>
Option sets the verbosity of report. Accepts a value from 0 to 9. Default value is 4.
<b>\$ tst .imply routine:[ name    glob ] .run [ path ]</b>
<b>\$ tst .imply r:[ name    glob ] .run [ path ]</b>
<b>\$ tst .run [ path ] routine:[ name    glob ]</b>
<b>\$ tst .run [ path ] r:[ name    glob ]</b>
Option to test separate test routine. Accepts name of test routine or glob.
<b>\$ tst .imply suite:[ name    glob ] .run [ path ]</b>
<b>\$ tst .run [ path ] suite:[ name    glob ]</b>
Option to test separate test suite. Accepts name of test routine or glob.
<b>\$ tst .imply testRoutineTimeOut:[ time ] .run [ path ]</b>
<b>\$ tst .run [ path ] testRoutineTimeOut:[ time ]</b>
Option limits the testing time for test routines. Accepts time in milliseconds. Default value is 5000ms.

<b>\$ tst .imply accuracy:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] accuracy:[ number ]</b>

Option sets the numeric deviation for the comparison of numerical values. Accepts numeric values of deviation. Default value is 1e-7.

## SECONDARY RUNNING OPTIONS

Running options that extend control of testing.

<b>\$ tst .imply sanitareTime:[ time ] .run [ path ]</b>
<b>\$ tst .run [ path ] sanitareTime:[ time ]</b>
Option sets the delay between completing the test suite and running the next one. Accepts time in milliseconds. Default value is 2000ms.
<b>\$ tst .imply negativity:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] negativity:[ number ]</b>
Option restricts the console output of passed routines and increases output of failed test checks. Accepts a value from 0 to 9. Default value is 1.
<b>\$ tst .imply silencing:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] silencing:[ number ]</b>
Option enables hiding the console output from the test object. Accepts 0 or 1. Default value is 0.
<b>\$ tst .imply shoulding:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] shoulding:[ number ]</b>
Option disables negative testing. Accepts 0 or 1. Default value is 0.
<b>\$ tst .imply fails:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] fails:[ number ]</b>
Option sets the number of errors received to interrupt the test. Accepts number of fails. By default is unlimited.
<b>\$ tst .imply beeping:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] beeping:[ number ]</b>
Option disables the beep after test completion. Accepts 0 or 1. Default value is 1.
<b>\$ tst .imply coloring:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] coloring:[ number ]</b>
Option makes report colourful. Accepts 0 or 1. Default value is 1.
<b>\$ tst .imply timing:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] timing:[ number ]</b>
Option disables measurement of time spent on testing. Accepts 0 or 1. Default value is 1.
<b>\$ tst .imply debug:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] debug:[ number ]</b>
Option sets value of Config.debug. Accepts 0 or 1. Default value is null, utility does not change debug mode of test object.
<b>\$ tst .imply rapidity:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] rapidity:[ number ]</b>
The option controls the amount of time spent on testing. Accepts values from -9 to +9. Default value is 0.
<b>\$ tst .imply concurrent:[ number ] .run [ path ]</b>
<b>\$ tst .run [ path ] concurrent:[ number ]</b>
Option enables parallel execution of test suites. Accepts 0 or 1. Default value is 0.

# TESTING CHEAT SHEET

## TEST SUITE STRUCTURE

A test file should contain only one test suite. Example of a minimum test file is given below. It uses the basic structural elements and can be considered as a test suite template.

dependencies

test routines definition

test suite definition

test suite launching

```
1
2 let _ = require( 'wTesting' );
3 let Join = require( './Join.js' );
4
5 //
6
7 function routine1( test )
8 {
9   test.identical( Join.join( 'Hello ', 'world!' ), 'Hello world!' );
10 }
11
12 //
13
14 function routine2( test )
15 {
16   test.case = 'pass';
17   test.identical( Join.join( 1, 3 ), '13' );
18
19   test.case = 'fail';
20   test.identical( Join.join( 1, 3 ), 13 );
21
22 }
23
24
25 //
26
27 var Self =
28 {
29   name : 'Join',
30   tests :
31   {
32     routine1,
33     routine2,
34   }
35 }
36
37 //
38
39 Self = wTestSuite( Self );
40 if( typeof module !== 'undefined' && !module.parent )
41   wTester.test( Self.name );
42
```

1st test routine

2nd test routine

test case

test case

test checks

test suite name

test suite routines

## TEST CHECKS

Test checks are the smallest structural element that checks one aspect of a test object behavior.

<b>test.is( boolLike arg );</b>
Passes if argument is true-like.
<b>test.isNot( boolLike arg );</b>
Passes if argument is false-like.
<b>test.identical( any arg1, any arg2 );</b> <b>test.il( any arg1, any arg2 );</b>
Passes if both arguments are identical.
<b>test.notIdentical( any arg1, any arg2 );</b> <b>test.ni( any arg1, any arg2 );</b>
Passes if both arguments are not identical.

<b>test.equivalent( any arg1, any arg2 );</b> <b>test.et( any arg1, any arg2 );</b>
Passes if both arguments are equivalent.
<b>test.notEquivalent( any arg1, any arg2 );</b> <b>test.ne( any arg1, any arg2 );</b>
Passes if both arguments are not equivalent.
<b>test.contains( any arg1, any arg2 );</b>
Passes if the arguments are identical or the first argument contains the second argument.
<b>test.gt( numberLike arg1, numberLike arg2 );</b>
Passes if the value of the first argument is greater than the value of the second.
<b>test.ge( numberLike arg1, numberLike arg2 );</b>
Passes if the value of the first argument is greater or equal to the value of the second.
<b>test.lt( numberLike arg1, numberLike arg2 );</b>
Passes if the value of the first argument is less than the value of the second.
<b>test.le( numberLike arg1, numberLike arg2 );</b>
Passes if the value of the first argument is less or equal to the value of the second.
<b>test.mustNotThrowError( routine arg );</b>
Passes if the routine does not throws an error synchronously or asynchronously.
<b>test.returnsSingleResource( routine arg );</b>
Passes if the routine ends synchronously or the consequence returns only one resource.
<b>test.shouldThrowErrorSync( routine arg );</b>
Passes if the routine throws an error synchronously.
<b>test.shouldThrowErrorAsync( routine arg );</b>
Passes if the routine throws an error asynchronously.
<b>test.shouldThrowErrorOfAnyKind( routine arg );</b>
Passes if the routine throws an error synchronously or asynchronously.