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.

```
$ npm install -g wTesting
```

Global installation of utility Testing by NPM.

MAIN COMMANDS

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

```
$ tst .help
```

Get help.

```
$ tst .help [ command ]
```

Get help on a specific command.

```
$ tst .suites.list [ path ]
```

Find test suites at a specific path.

```
$ tst .run [ path ]
```

Run test suites found at a specific path.

```
$ tst .imply [ options... ] .run [ path ]
```

Change state or imply variable value.

RUNNING TESTS

```
$ tst .run [ path to a test file ( directory ) || path with glob ]
$ node [ path to a test file ]
```

Running of test suite (test suites) by utility Testing and NodeJS.

PRIMARY RUNNING OPTIONS

To control testing the running options is used.

```
$ tst .imply verbosity:[ number ] .run [ path ]
$ tst .imply v:[ number ] .run [ path ]
$ tst .run [ path ] verbosity:[ number ]
$ tst .run [ path ] v:[ number ]
```

Option sets the verbosity of report. Accepts a value from 0 to 9. Default value is 4.

```
$ tst .imply routine:[ name ] .run [ path ]
$ tst .imply r:[ name ] .run [ path ]
$ tst .run [ path ] routine:[ name ]
$ tst .run [ path ] r:[ name ]
```

Option to test separate test routine. Accepts name of test routine.

```
$ tst .imply testRoutineTimeOut:[ time ] .run [ path ]
$ tst .run [ path ] testRoutineTimeOut:[ time ]
```

Option limits the testing time for test routines. Accepts time in milliseconds. Default value is 5000ms.

```
$ tst .imply accuracy:[ number ] .run [ path ]
$ tst .run [ path ] accuracy:[ number ]
```

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.

```
$ tst .imply sanitareTime:[ time ] .run [ path ]
$ tst .run [ path ] sanitareTime:[ time ]
```

Option sets the delay between completing the test suite and running the next one. Accepts time in milliseconds. Default value is 2000ms.

```
$ tst .imply importanceOfNegative:[ number ] .run [ path ]
$ tst .run [ path ] importanceOfNegative:[ number ]
```

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.

```
$ tst .imply silencing:[ number ] .run [ path ]
$ tst .run [ path ] silencing:[ number ]
```

Option enables hiding the console output from the test object. Accepts 0 or 1. Default value is 0.

```
$ tst .imply shoulding:[ number ] .run [ path ]
$ tst .run [ path ] shoulding:[ number ]
```

Option disables negative testing. Accepts 0 or 1. Default value is 0.

```
$ tst .imply fails:[ number ] .run [ path ]
$ tst .run [ path ] fails:[ number ]
```

Option sets the number of errors received to interrupt the test. Accepts number of fails. By default is unlimited.

```
$ tst .imply beeping:[ number ] .run [ path ]
$ tst .run [ path ] beeping:[ number ]
```

Option disables the beep after test completion. Accepts 0 or 1. Default value is 1.

```
$ tst .imply coloring:[ number ] .run [ path ]
$ tst .run [ path ] coloring:[ number ]
```

Option makes report colourful. Accepts 0 or 1. Default value is 1.

```
$ tst .imply timing:[ number ] .run [ path ]
$ tst .run [ path ] timing:[ number ]
```

Option disables measurement of time spent on testing. Accepts 0 or 1. Default value is 1.

```
$ tst .imply rapidity:[ number ] .run [ path ]
$ tst .run [ path ] rapidity:[ number ]
```

The option controls the amount of time spent on testing. Accepts values from -9 to +9. Default value is 0.

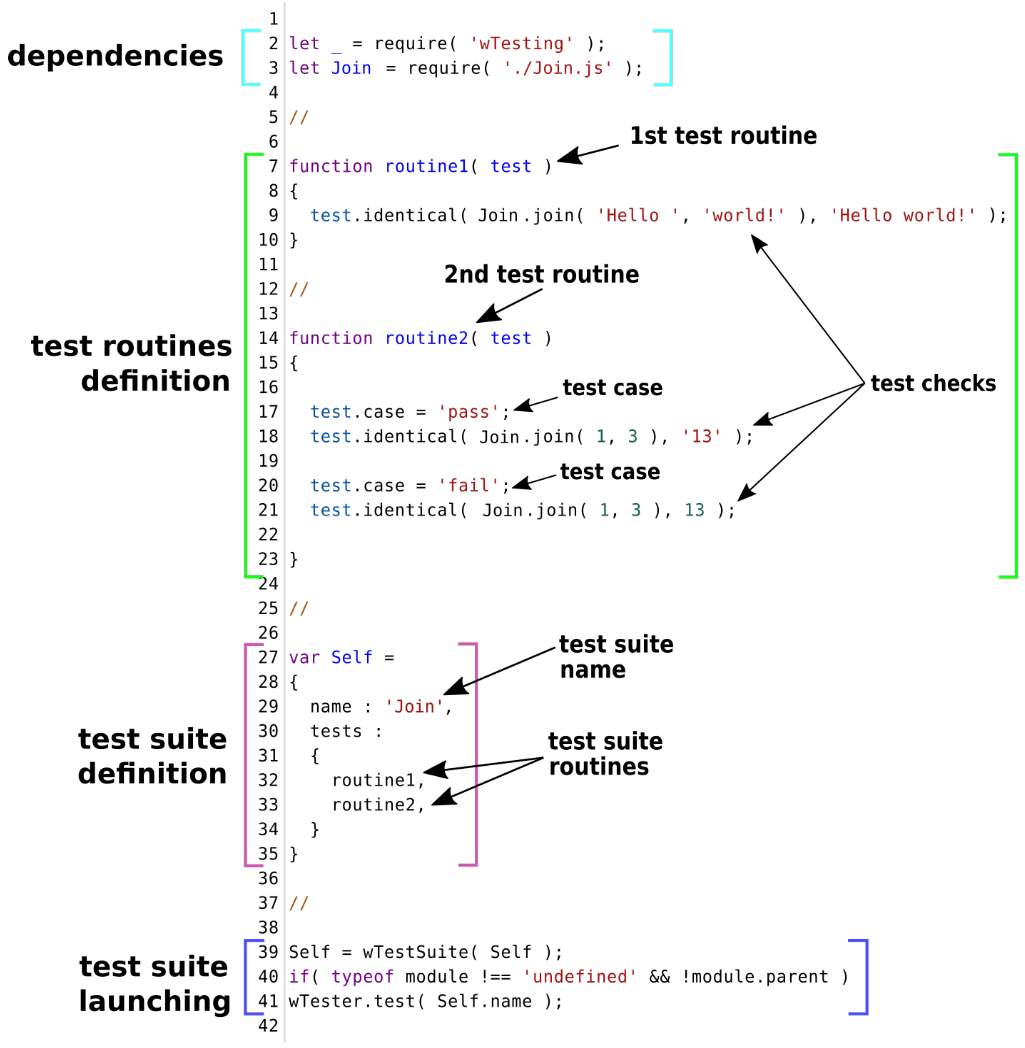
```
$ tst .imply  concurrent:[ number ] .run [ path ]
$ tst .run [ path ] concurrent:[ number ]
```

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.



```
let _ = require( 'wTesting' );
let Join = require( './Join.js' );

//

function routine1( test )
{
  test.identical( Join.join( 'Hello ', 'world!' ), 'Hello world!' );
}

//

function routine2( test )
{

  test.case = 'pass';
  test.identical( Join.join( 1, 3 ), '13' );

  test.case = 'fail';
  test.identical(  Join.join( 1, 3 ), 13 );

}

//

var Self =
{
  name : 'Join',
  tests :
  {
    routine1,
    routine2,
  }
}

//

Self = wTestSuite( Self );
if( typeof module !== 'undefined' && !module.parent )
wTester.test( Self.name );
```

TEST CHECKS

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

is(boolLike arg);

Passes if argument is true-like.

isNot(boolLike arg);

Passes if argument is false-like.

isNotError(errorLike arg);

Passes if argument is not error.

identical(any arg1, any arg2);
il(any arg1, any arg2);

Passes if both arguments are identical.

notIdentical(any arg1, any arg2);
ni(any arg1, any arg2);

Passes if both arguments are not identical.

equivalent(any arg1, any arg2);
et(any arg1, any arg2);

Passes if both arguments are equivalent.

notEquivalent(any arg1, any arg2);
ne(any arg1, any arg2);

Passes if both arguments are not equivalent.

contains(any arg1, any arg2);

Passes if the arguments are identical or the first argument contains the second argument.

setsAreIdentical(arrayLike arg1, arrayLike arg2);

Passes if elements of both arguments are identical.

gt(numberLike arg1, numberLike arg2);

Passes if the value of the first argument is greater than the value of the second.

ge(numberLike arg1, numberLike arg2);

Passes if the value of the first argument is greater or equal to the value of the second.

lt(numberLike arg1, numberLike arg2);

Passes if the value of the first argument is less than the value of the second.

le(numberLike arg1, numberLike arg2);

Passes if the value of the first argument is less or equal to the value of the second.

mustNotThrowError(routine arg);

Passes if the routine does not throws an error synchronously or asynchronously.

shouldMessageOnlyOnce(routine arg);

Passes if the routine ends synchronously or the concequence returns only one resource.

shouldThrowErrorSync(routine arg);

Passes if the routine throws an error synchronously.

shouldThrowErrorAsync(routine arg);

Passes if the routine throws an error asynchronously.

shouldThrowError(routine arg);

Passes if the routine throws an error synchronously or asynchronously.