

## Simple Use Case: TestCase Object

Create a TestCase object with desired tests, add your TestCase to the TestRunner object, and run the test:

```
//set up a test case:
var oTestCase = new YAHOO.tool.TestCase({
  name: "Simple Math",
  testEquality: function () {
    YAHOO.util.Assert.areEqual(4, (2+2), "2+2=4");
  };
});
//add the test case to the TestRunner:
YAHOO.tool.TestRunner.add(oTestCase);
YAHOO.tool.TestRunner.run(); //run the test
```

## Key Members of the TestCase Object

|                 |   |               |   |              |  |
|-----------------|---|---------------|---|--------------|--|
| <b>name</b>     | The name of the TestCase. This will be visible in the logging of TestCase events.   |               |   |              |  |
| <b>testname</b> | A function that tests functional code via one or more assertions; name must begin with the string "test". A TestCase can have one or more test members.   |               |   |              |  |
| <b>setUp</b>    | Method that prepares your test case to run by, for example, creating needed data constructs or objects.   |               |   |              |  |
| <b>tearDown</b> | Method that nulls out variables in use by the TestCase, detaches event handlers, etc.   |               |   |              |  |
| <b>_should</b>  | Special object that provides granular configuration of the test case. It can have the following members: <table border="1" style="width: 100%; margin-top: 10px;"> <tr> <td><b>ignore</b></td> <td>A name:value pair consisting of a testname and Boolean indicating whether to ignore the test (e.g.: <code>ignore: {testOne : true /*ignore testOne*/}</code>).</td> </tr> <tr> <td><b>error</b></td> <td>A name:value pair consisting of a testname and Boolean indicating whether the test is should fail:<br/><br/> <code>error: {testTwo : true /*testTwo should fail*/}</code><br/><br/>           Or, a specific error string can substitute for the Boolean:<br/><br/> <code>error: {testTwo : "Expected string." /*testTwo should fail with this specific error message*/}</code>.         </td> </tr> </table> | <b>ignore</b> | A name:value pair consisting of a testname and Boolean indicating whether to ignore the test (e.g.: <code>ignore: {testOne : true /*ignore testOne*/}</code> ). | <b>error</b> | A name:value pair consisting of a testname and Boolean indicating whether the test is should fail:<br><br><code>error: {testTwo : true /*testTwo should fail*/}</code><br><br>Or, a specific error string can substitute for the Boolean:<br><br><code>error: {testTwo : "Expected string." /*testTwo should fail with this specific error message*/}</code> . |
| <b>ignore</b>   | A name:value pair consisting of a testname and Boolean indicating whether to ignore the test (e.g.: <code>ignore: {testOne : true /*ignore testOne*/}</code> ).   |               |   |              |  |
| <b>error</b>    | A name:value pair consisting of a testname and Boolean indicating whether the test is should fail:<br><br><code>error: {testTwo : true /*testTwo should fail*/}</code><br><br>Or, a specific error string can substitute for the Boolean:<br><br><code>error: {testTwo : "Expected string." /*testTwo should fail with this specific error message*/}</code> .  |               |   |              |  |

All TestCase configurations should be passed into the TestCase constructor as an object literal; see Simple Use Case.

## Key Events in YUITest

See online docs for full list of custom events, including TestSuite- and TestRunner-level events.

All YUITest events are subscribed to at the TestRunner level; e.g.:  
`YAHOO.tool.TestRunner.subscribe(YAHOO.tool.TestRunner.TEST_FAIL_EVENT, myFn);`

| Test-level Events                                      |   |                              |  |
|--|---|------------------------------|--|
| Event:   | Fires:  |                              |  |
| TEST_PASS_EVENT  | When an individual test passes.                 |                              |  |
| TEST_FAIL_EVENT  | When an individual test fails.                  |                              |  |
| <i>Argument Data Object for Test-level Events:</i>     |   |                              |  |
| <b>type</b>  | Type of event.                                  |                              |  |
| <b>testCase</b>  | The testCase object to which this test belongs. |                              |  |
| <b>testName</b>  | The string name of this test.                   |                              |  |
| TestCase-level Events                                  |   |                              |  |
| Event:   | Fires:  |                              |  |
| TEST_CASE_BEGIN_EVENT                                  | Before the TestCase is run.                     |                              |  |
| TEST_CASE_COMPLETE_EVENT                               | After the TestCase is run.                      |                              |  |
| <i>Argument Data Object for TestCase-level Events:</i> |   |                              |  |
| <b>type</b>  | Type of event.                                  |                              |  |
| <b>testCase</b>  | Current TestCase instance.                      |                              |  |
| <b>results</b> (TEST_CASE_END event only)              | <b>passed</b>                                   | Number of tests that passed. |  |
|  | <b>failed</b>                                   | Number of tests that failed. |  |
|  | <b>testname</b>                                 | <b>result</b>                | <code>pass</code> or <code>fail</code> . |
|  |   | <b>message</b>               | String returned by the test.             |

**Note:** TestSuite- and TestRunner-level events are also available, containing summary data in addition to specific TestCase results objects. See online docs for full details.

## Assertions

Assertions are accessed via `YAHOO.util.Assert`

|   |   |
|---|---|
| <p><b>Equality assertions:</b><br/> <code>areEqual(expected, actual)</code><br/> <code>areNotEqual(expected, actual)</code><br/>           equivalent to <code>== test</code></p> <p><b>Sameness assertions:</b><br/> <code>areSame(expected, actual)</code><br/> <code>areNotSame(expected, actual)</code><br/>           equivalent to <code>=== test</code></p> <p><b>Data-type assertions:</b><br/> <code>isArray(arg)</code><br/> <code>isBoolean(arg)</code><br/> <code>isFunction(arg)</code><br/> <code>isNumber(arg)</code><br/> <code>isObject(arg)</code> <i>object and function return true</i><br/> <code>isString(arg)</code></p> | <p><b>isTypeOf assertion:</b><br/> <code>isTypeOf(sType, sTest, sFailureMessage)</code><br/> <code>YAHOO.util.Assert.isTypeOf("string", 5, "Expected string.");</code><br/> <i>//fails</i></p> <p><b>isInstanceOf assertion:</b><br/> <code>isInstanceOf(oConstructor, oTestObject, sFailureMessage)</code><br/> <code>YAHOO.util.Assert.isInstanceOf(String, "Madrone", "Expected string.");</code><br/> <i>//passes</i><br/> <i>can be used to test non-native objects, too</i></p> <p><b>Special Value Assertions:</b><br/> <code>isFalse, isTrue, isNaN, isNotNaN, isNull, isNotNull, isUndefined, isNotUndefined</code><br/> <code>YAHOO.util.Assert.isNull(7, "Expected null.");</code><br/> <i>//fails</i></p> |
|---|---|

## YAHOO.tool.TestSuite

Methods:

**add(obj testCase or testSuite)** adds a testCase or testSuite object to a testSuite

## YAHOO.tool.TestRunner

Methods:

**add(obj testCase or testSuite)** adds a testCase or testSuite object to the list of items to run

**clear()** removes all test objects from the runner

**run()** runs all testCase and testSuites currently queued in the TestRunner

## YAHOO.tool.TestCase

Methods:

**wait([fn segment, int delay])** causes the TestCase to delay the specified number of milliseconds before *segment* is executed

**resume([fn segment])** resumes a paused test; if *segment* is omitted, the test automatically passes

## YAHOO.util.UserAction

Methods:

**click(obj target, obj options)** simulates a click on the target

**mousedown(obj target, obj options)** simulates a mousedown on the target

**mouseup(obj target, obj options)** simulates a mouseup on the target

**mouseover(obj target, obj options)** simulates a mouseover on the target

**mouseout(obj target, obj options)** simulates a mouseout on the target

**mousemove(obj target, obj options)** simulates a mousemove on the target

**keydown(obj target, obj options)** simulates a keydown on the target

**keyup(obj target, obj options)** simulates a keyup on the target

**keypress(obj target, obj options)** simulates a keypress on the target

## Dependencies

The YUI Test Utility requires the Yahoo Global Object, Dom Collection, Event Utility, and Logger Control. The Logger's CSS file and the YUI Test CSS file are also required.