CPPTEST CHEATSHEET `#include "cpptest.h"`

Test::Suite

```
Suite ()
virtual ~Suite ()
void add (std::auto_ptr< Suite > suite)
bool run (Output &output, bool cont_after_fail=true)
```

Test::CompilerOutput

```
enum Format { Generic, BCC, GCC, MSVC }

CompilerOutput (Format format=Generic, std::ostream &stream=std::cout)
CompilerOutput (const std::string &format, std::ostream &stream=std::cout)
virtual void assertment (const Source &s)
```

Test::TextOutput

```
enum Mode { Terse, Verbose }

TextOutput (Mode mode, std::ostream &stream=std::cout)
virtual void finished (int tests, const Time &time)
virtual void suite_start (int tests, const std::string &name)
virtual void suite_end (int tests, const std::string &name, const Time &time)
virtual void test_end (const std::string &name, bool ok, const Time &time)
virtual void assertment (const Source &s)
```

Test::HtmlOutput

```
void generate (std::ostream &os, bool incl_ok_tests=true, const std::string &name="")
```

Example Fixture

```
class SomeTestSuite: public Test::Suite
{
public:
    SomeTestSuite()
    {
        TEST_ADD(SomeTestSuite::test1)
        TEST_ADD(SomeTestSuite::test2)
    }
protected:
    // setup resources...
    virtual void setup()     {}
    // remove resources...
    virtual void tear_down() {}
private:
    // use common resources...
    void test1() {}
    // use common resources...
    void test2() {}
};
```

Example Suite

```
// ... with many tests
class TestSuite1: public Test::Suite { };
// ... with many tests
class TestSuite2: public Test::Suite { };
// ... with many tests
class TestSuite3: public Test::Suite { };

bool run_tests()
{
    Test::Suite ts;
    ts.add(auto_ptr<Test::Suite>(new TestSuite1));
    ts.add(auto_ptr<Test::Suite>(new TestSuite2));
    ts.add(auto_ptr<Test::Suite>(new TestSuite3));

    Test::TextOutput output(Test::TextOutput::Verbose);
    return ts.run(output);
}
```

General asserts

```
TEST_FAIL(msg)
```

Comparision asserts

```
TEST_ASSERT(expr)
TEST_ASSERT_MSG(expr, msg)
TEST_ASSERT_DELTA(a, b, delta)
TEST_ASSERT_DELTA_MSG(a, b, delta, msg)
```

Exception asserts

```
TEST_THROWS(expr, x)
TEST_THROWS_MSG(expr, x, msg)
TEST_THROWS_ANYTHING(expr)
TEST_THROWS_ANYTHING_MSG(expr, msg)
TEST_THROWS_NOTHING(expr)
TEST_THROWS_NOTHING_MSG(expr, msg)
```