Visual Testing using PhantomCSS in JavaScript

Overview

In this tutorial, we will create some tests in JavaScript using PhantomCSS.

> **Please note**
>
> This tutorial has been tested with the v0 branch of PhantomCSS. The master branch is not actively maintained and may not be stable and compatible with the latest version of PhantomJS.

> **Please note**
>
> As of 2019, CasperJS seems to be no longer maintained.

Requirements

- phantomjs
- casperjs
- resemblejs
- phantomCSS

Description

First, please ensure your Tests are clearly identified in the "pass" and "fail" callbacks. This can be done by editing the phantomcss.js source file or by defining the callbacks during the PhantomCSS initialization.

```
function _onPass( test ) {
        console.log( '\n' );
        var name = 'Should look the same ' + test.filename;
        casper.test.pass(name, {name: name});
}

function _onFail( test ) {
        console.log('\n');
        var name = 'Should look the same ' + test.filename;
        casper.test.fail(name, {name:name, message: 'Looks different (' + test.mismatch + '% mismatch) ' + test.
failFile });
}
```

The JavaScript containing the visual assertions is under the demo/testsuite.js source file found in PhantomCSS's repository.

demo/testsuite.js (from the demo of PhantomCSS)

```
/*
        Require and initialise PhantomCSS module
        Paths are relative to CasperJs directory
*/
var fs = require( 'fs' );
var path = fs.absolute( fs.workingDirectory + '/phantomcss.js' );
var phantomcss = require( path );
var server = require('webserver').create();
var html = fs.read( fs.absolute( fs.workingDirectory + '/demo/coffeemachine.html' ));
server.listen(8080,function(req,res){
```

```
        res.statusCode = 200;
        res.headers = {
                'Cache': 'no-cache',
                'Content-Type': 'text/html;charset=utf-8'
        };
        res.write(html);
        res.close();
});

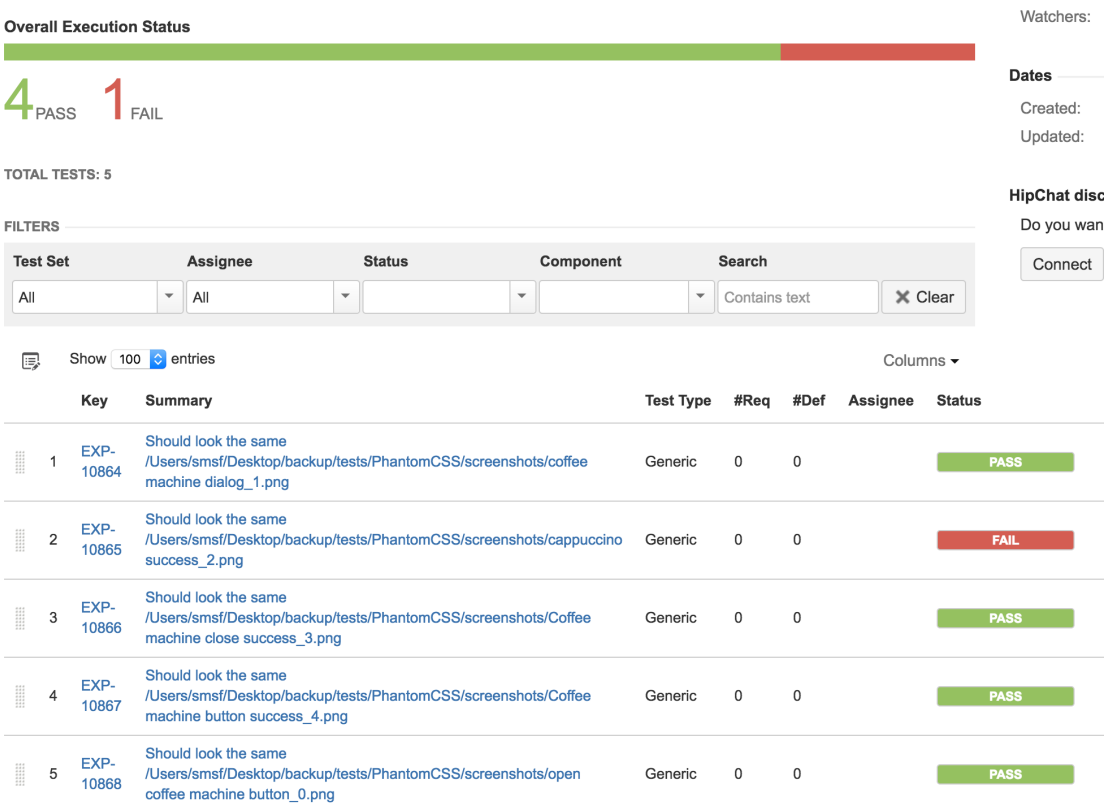
casper.test.begin( 'Coffee machine visual tests', function ( test ) {
        phantomcss.init( {
                rebase: casper.cli.get( "rebase" ),
                // SlimerJS needs explicit knowledge of this Casper, and lots of absolute paths
                casper: casper,
                libraryRoot: fs.absolute( fs.workingDirectory + '' ),
                screenshotRoot: fs.absolute( fs.workingDirectory + '/screenshots' ),
                failedComparisonsRoot: fs.absolute( fs.workingDirectory + '/demo/failures' ),
                addLabelToFailedImage: false,
                /*
                screenshotRoot: '/screenshots',
                failedComparisonsRoot: '/failures'
                casper: specific_instance_of_casper,
                libraryRoot: '/phantomcss',
                fileNameGetter: function overide_file_naming(){},
                onPass: function passCallback(){},
                onFail: function failCallback(){},
                onTimeout: function timeoutCallback(){},
                onComplete: function completeCallback(){},
                hideElements: '#thing.selector',
                addLabelToFailedImage: true,
                outputSettings: {
                        errorColor: {
                                red: 255,
                                green: 255,
                                blue: 0
                        },
                        errorType: 'movement',
                        transparency: 0.3
                }*/
        } );
        casper.on( 'remote.message', function ( msg ) {
                this.echo( msg );
        } );
        casper.on( 'error', function ( err ) {
                this.die( "PhantomJS has errored: " + err );
        } );
        casper.on( 'resource.error', function ( err ) {
                casper.log( 'Resource load error: ' + err, 'warning' );
        } );
        /*
                The test scenario
        */
        casper.start( 'http://localhost:8080' );
        casper.viewport( 1024, 768 );
        casper.then( function () {
                phantomcss.screenshot( '#coffee-machine-wrapper', 'open coffee machine button' );
        } );
        casper.then( function () {
                casper.click( '#coffee-machine-button' );
                // wait for modal to fade-in
                casper.waitForSelector( '#myModal:not([style*="display: none"])',
                        function success() {
                                phantomcss.screenshot( '#myModal', 'coffee machine dialog' );
                        },
                        function timeout() {
                                casper.test.fail( 'Should see coffee machine' );
                        }
                );
        } );
        casper.then( function () {
                casper.click( '#cappuccino-button' );
```

```
                phantomcss.screenshot( '#myModal', 'cappuccino success' );
        } );
        casper.then( function () {
                casper.click( '#close' );
                // wait for modal to fade-out
                casper.waitForSelector( '#myModal[style*="display: none"]',
                        function success() {
                                phantomcss.screenshot( {
                                        'Coffee machine close success': {
                                                selector: '#coffee-machine-wrapper',
                                                ignore: '.selector'
                                        },
                                        'Coffee machine button success': '#coffee-machine-button'
                                } );
                        },
                        function timeout() {
                                casper.test.fail( 'Should be able to walk away from the coffee machine' );
                        }
                );
        } );
        casper.then( function now_check_the_screenshots() {
                // compare screenshots
                phantomcss.compareAll();
        } );
        /*
        Casper runs tests
        */
        casper.run( function () {
                console.log( '\nTHE END.' );
                // phantomcss.getExitStatus() // pass or fail?
                casper.test.done();
        } );
} );
```

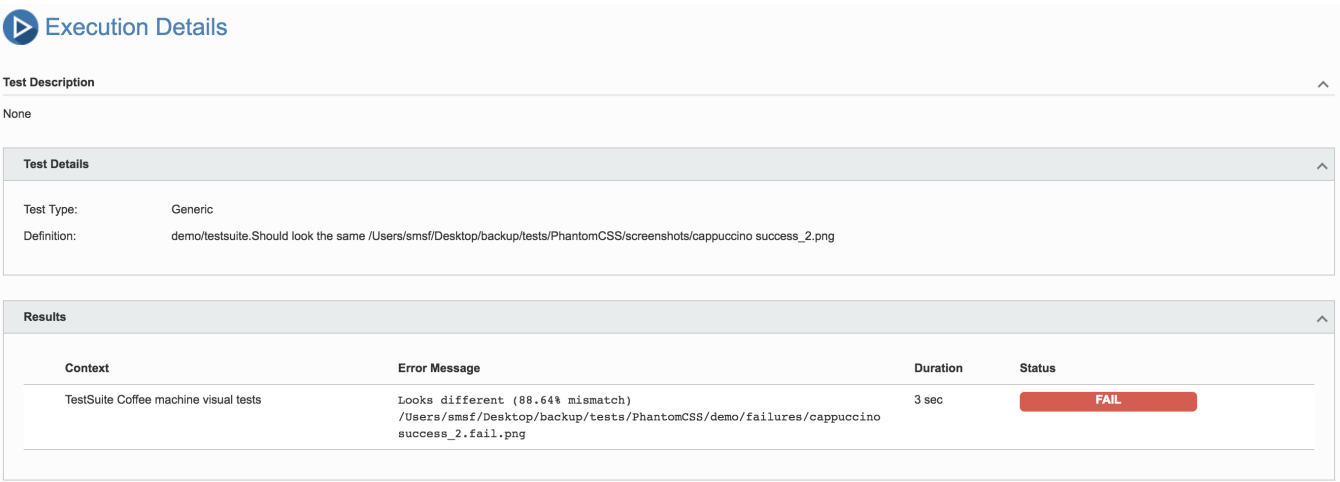
After running the tests and generating the JUnit XML report (e.g., vlog.xml), it can be imported to Xray (either by the REST API or through the **Import Execution Results** action within the Test Execution).

```
casperjs test demo/testsuite.js --xunit=vlog.xml
```

JUnit's Test Case is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains the base name of the source file concatenated with the name of file being validated.



The Execution Details of the Generic Test contains information about the Test Suite, which in this case corresponds to "Coffee machine visual tests".



References

- https://github.com/Huddle/PhantomCSS
- https://github.com/Huddle/PhantomCSS/tree/v0/demo
- ~~http://docs.casperjs.org~~