```
 *

 */


/*global process, console, phantom, slimer, require:true*/

/*jshint maxstatements:34, maxcomplexity:10*/


// node check

if ('process' in this && process.title === "node") {

   console.error('CasperJS cannot be executed within a nodejs environment');

   process.exit(1);

}


// phantom check

if (!('phantom' in this)) {

   console.error('CasperJS needs to be executed in a PhantomJS environment
http://phantomjs.org/');

}


// Common polyfills

if (typeof Function.prototype.bind !== "function") {

   // https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Reference/Global_Objects/Function/bind#Compatibility

   Function.prototype.bind = function (oThis) {

      "use strict";

      /* jshint -W055 */

      if (typeof this !== "function") {

         // closest thing possible to the ECMAScript 5 internal IsCallable function

         throw new TypeError("Function.prototype.bind - what is trying to be bound is not
callable");
```

```
        }
        var aArgs = Array.prototype.slice.call(arguments, 1),
            fToBind = this,
            fNOP = function() {},
            fBound = function() {
              return fToBind.apply(this instanceof fNOP && oThis ? this : oThis,
                                   aArgs.concat(Array.prototype.slice.call(arguments)));
            };
        fNOP.prototype = this.prototype;
        fBound.prototype = new fNOP();
        return fBound;
    };
}

// Custom base error
var CasperError = function CasperError(msg) {
    "use strict"
    Error.call(this);
    this.message = msg;
    this.name = 'CasperError';
};
CasperError.prototype = Object.getPrototypeOf(new Error());

// casperjs env initialization
(function(global, phantom){
    /*jshint maxstatements:99*/
    "use strict";
```

```
// phantom args
var sysargs = require('system').args;
var phantomArgs = [];
// don't take the first value; this mimics the contents of the old phantom.args
for (var i = 1; i < sysargs.length; i++) {
    phantomArgs.push(sysargs[i]);
}

if (phantom.casperLoaded) {
    return;
}

function __die(message) {
    if (message) {
        console.error(message);
    }
    phantom.exit(1);
}

function __terminate(message) {
    if (message) {
        console.log(message);
    }
    phantom.exit();
}

(function (version) {
```

```
    // required version check
    if (version.major === 1) {
        if (version.minor < 8) {
            return __die('CasperJS needs at least PhantomJS v1.8 or later.');
        }
        if (version.minor === 8 && version.patch < 1) {
            return __die('CasperJS needs at least PhantomJS v1.8.1 or later.');
        }
    } else if (version.major === 2) {
        console.log("Warning PhantomJS v2.0 not yet released. There will not be any official support for any bugs until stable version is released!");
    }
    else return __die('CasperJS needs PhantomJS v1.x or v2.x');
})(phantom.version);

// Hooks in default phantomjs error handler
phantom.onError = function onPhantomError(msg, trace) {
    phantom.defaultErrorHandler.apply(phantom, arguments);
    // print a hint when a possible casperjs command misuse is detected
    if (msg.indexOf("ReferenceError: Can't find variable: casper") === 0) {
        console.error('Hint: you may want to use the `casperjs test` command.');
    }
    // exits on syntax error
    if (msg.indexOf('SyntaxError: Parse error') === 0) {
        __die();
    }
};
```

```
// Patching fs
var fs = (function patchFs(fs) {
   if (!fs.hasOwnProperty('basename')) {
      fs.basename = function basename(path) {
         return path.replace(/.*\//, '');
      };
   }
   if (!fs.hasOwnProperty('dirname')) {
      fs.dirname = function dirname(path) {
         if (!path) return undefined;
         return path.toString().replace(/\\/g, '/').replace(/\/[^\/]*$/, '');
      };
   }
   if (!fs.hasOwnProperty('isWindows')) {
      fs.isWindows = function isWindows() {
         var testPath = arguments[0] || this.workingDirectory;
         return (/^[a-z]{1,2}:/i).test(testPath) || testPath.indexOf("\\\\") === 0;
      };
   }
   if (fs.hasOwnProperty('joinPath')) {
      fs.pathJoin = fs.joinPath;
   } else if (!fs.hasOwnProperty('pathJoin')) {
      fs.pathJoin = function pathJoin() {
         return Array.prototype.join.call(arguments, '/');
      };
   }
   return fs;
```

```
})(require('fs'));


// CasperJS root path
if (!phantom.casperPath) {
    try {
        phantom.casperPath = phantomArgs.map(function _map(arg) {
            var match = arg.match(/^--casper-path=(.*)/);
            if (match) {
                return fs.absolute(match[1]);
            }
        }).filter(function _filter(path) {
            return fs.isDirectory(path);
        }).pop();
    } catch (e) {
        return __die("Couldn't find nor compute phantom.casperPath, exiting.");
    }
}


/**
 * Prints CasperJS help.
 */
function printHelp() {
    var engine = phantom.casperEngine === 'slimerjs' ? slimer : phantom;
    var version = [engine.version.major, engine.version.minor, engine.version.patch].join('.');
    return __terminate([
        'CasperJS version ' + phantom.casperVersion.toString() +
        ' at ' + phantom.casperPath + ', using ' + phantom.casperEngine + ' version ' + version,
```

```
        fs.read(fs.pathJoin(phantom.casperPath, 'bin', 'usage.txt'))
    ].join('\n'))
}

/**
 * Patched require to allow loading of native casperjs modules.
 * Every casperjs module have to first call this function in order to
 * load a native casperjs module:
 *
 *     var require = patchRequire(require);
 *     var utils = require('utils');
 *
 * Useless for SlimerJS
 */
function patchRequire(require) {
    if (require.patched) {
        return require;
    }
    function fromPackageJson(module, dir) {
        var pkgPath, pkgContents, pkg;
        pkgPath = fs.pathJoin(dir, module, 'package.json');
        if (!fs.exists(pkgPath)) {
            return;
        }
        pkgContents = fs.read(pkgPath);
        if (!pkgContents) {
            return;
```

```
    }
    try {
        pkg = JSON.parse(pkgContents);
    } catch (e) {
        return;
    }
    if (typeof pkg === "object" && pkg.main) {
        return fs.absolute(fs.pathJoin(dir, module, pkg.main));
    }
}
function resolveFile(path, dir) {
    var extensions = ['js', 'coffee', 'json'];
    var basenames = [path, path + '/index'];
    var paths = [];
    var nodejsScript = fromPackageJson(path, dir);
    if (nodejsScript) {
        return nodejsScript;
    }
    basenames.forEach(function(basename) {
        paths.push(fs.absolute(fs.pathJoin(dir, basename)));
        extensions.forEach(function(extension) {
            paths.push(fs.absolute(fs.pathJoin(dir, [basename, extension].join('.'))));
        });
    });
    for (var i = 0; i < paths.length; i++) {
        if (fs.isFile(paths[i])) {
            return paths[i];
```

```
        }
    }
    return null;
}
function getCurrentScriptRoot() {
    if ((phantom.casperScriptBaseDir || "").indexOf(fs.workingDirectory) === 0) {
        return phantom.casperScriptBaseDir;
    }
    return fs.absolute(fs.pathJoin(fs.workingDirectory, phantom.casperScriptBaseDir));
}
function casperBuiltinPath(path) {
    return resolveFile(path, fs.pathJoin(phantom.casperPath, 'modules'));
}
function nodeModulePath(path) {
    var resolved, prevBaseDir;
    var baseDir = getCurrentScriptRoot();
    do {
        resolved = resolveFile(path, fs.pathJoin(baseDir, 'node_modules'));
        prevBaseDir = baseDir;
        baseDir = fs.absolute(fs.pathJoin(prevBaseDir, '..'));
    } while (!resolved && baseDir !== '/' && baseDir !== prevBaseDir);
    return resolved;
}
function localModulePath(path) {
    return resolveFile(path, phantom.casperScriptBaseDir || fs.workingDirectory);
}
var patchedRequire = function patchedRequire(path) {
```

```
        try {
            return require(casperBuiltinPath(path) ||
                           nodeModulePath(path)    ||
                           localModulePath(path)   ||
                           path);
        } catch (e) {
            throw new CasperError("Can't find module " + path);
        }
    };
    patchedRequire.cache = require.cache;
    patchedRequire.extensions = require.extensions;
    patchedRequire.stubs = require.stubs;
    patchedRequire.patched = true;
    return patchedRequire;
}

/**
 * Initializes the CasperJS Command Line Interface.
 */
function initCasperCli(casperArgs) {
    /* jshint maxcomplexity:99 */
    var baseTestsPath = fs.pathJoin(phantom.casperPath, 'tests');

    function setScriptBaseDir(scriptName) {
        var dir = fs.dirname(scriptName);
        if (dir === scriptName) {
            dir = '.';
```

```
        }

        phantom.casperScriptBaseDir = dir;
    }


    if (!!casperArgs.options.version) {
        return __terminate(phantom.casperVersion.toString())
    } else if (casperArgs.get(0) === "test") {
        phantom.casperScript = fs.absolute(fs.pathJoin(baseTestsPath, 'run.js'));
        phantom.casperTest = true;
        casperArgs.drop("test");
        setScriptBaseDir(casperArgs.get(0));
    } else if (casperArgs.get(0) === "selftest") {
        phantom.casperScript = fs.absolute(fs.pathJoin(baseTestsPath, 'run.js'));
        phantom.casperSelfTest = phantom.casperTest = true;
        casperArgs.options.includes = fs.pathJoin(baseTestsPath, 'selftest.js');
        if (casperArgs.args.length <= 1) {
            casperArgs.args.push(fs.pathJoin(baseTestsPath, 'suites'));
        }
        casperArgs.drop("selftest");
        phantom.casperScriptBaseDir = fs.dirname(casperArgs.get(1) ||
fs.dirname(phantom.casperScript));
    } else if (casperArgs.args.length === 0 || !!casperArgs.options.help) {
        return printHelp();
    }


    if (!phantom.casperScript) {
        phantom.casperScript = casperArgs.get(0);
    }
```

```
    if (phantom.casperScript !== "/dev/stdin" && !fs.isFile(phantom.casperScript)) {
        return __die('Unable to open file: ' + phantom.casperScript);
    }

    if (!phantom.casperScriptBaseDir) {
        setScriptBaseDir(phantom.casperScript);
    }

    // filter out the called script name from casper args
    casperArgs.drop(phantom.casperScript);
}

// CasperJS version, extracted from package.json - see http://semver.org/
phantom.casperVersion = (function getCasperVersion(path) {
    var parts, patchPart, pkg, pkgFile;
    pkgFile = fs.absolute(fs.pathJoin(path, 'package.json'));
    if (!fs.exists(pkgFile)) {
        throw new CasperError('Cannot find package.json at ' + pkgFile);
    }
    try {
        pkg = JSON.parse(require('fs').read(pkgFile));
    } catch (e) {
        throw new CasperError('Cannot read package file contents: ' + e);
    }
    parts  = pkg.version.trim().split(".");
    if (parts.length < 3) {
```

```
        throw new CasperError("Invalid version number");
    }
    patchPart = parts[2].split('-');
    return {
        major: ~~parts[0]      || 0,
        minor: ~~parts[1]      || 0,
        patch: ~~patchPart[0]  || 0,
        ident: patchPart[1]    || "",
        toString: function toString() {
            var version = [this.major, this.minor, this.patch].join('.');
            if (this.ident) {
                version = [version, this.ident].join('-');
            }
            return version;
        }
    };
})(phantom.casperPath);

if ("paths" in global.require) {
    // declare a dummy patchRequire function
    global.patchRequire = function(req) {return req;};

    require.paths.push(fs.pathJoin(phantom.casperPath, 'modules'));
    require.paths.push(fs.workingDirectory);
} else {
    global.__require = require;
    global.patchRequire = patchRequire; // must be called in every casperjs module as of 1.1
```

```
    global.require = patchRequire(global.require);
}

if ("slimer" in global) {
    require.globals.patchRequire = global.patchRequire;
    require.globals.CasperError = CasperError;
    phantom.casperEngine = "slimerjs";
} else {
    phantom.casperEngine = "phantomjs";
}

// casper cli args
phantom.casperArgs = require('cli').parse(phantomArgs);

if (true === phantom.casperArgs.get('cli')) {
    initCasperCli(phantom.casperArgs);
}

if ("paths" in global.require) {
    if ((phantom.casperScriptBaseDir || "").indexOf(fs.workingDirectory) === 0) {
        require.paths.push(phantom.casperScriptBaseDir);
    } else {
        require.paths.push(fs.pathJoin(fs.workingDirectory, phantom.casperScriptBaseDir));
    }
    require.paths.push(fs.pathJoin(require.paths[require.paths.length-1], 'node_modules'));
}
```

```
    // casper loading status flag
    phantom.casperLoaded = true;

    // passed casperjs script execution
    if (phantom.casperScript && !phantom.injectJs(phantom.casperScript)) {
        return __die('Unable to load script ' + phantom.casperScript + '; check file syntax');
    }
})(this, phantom);
```