Karma Testing Cheatsheet

A comprehensive cheat sheet for Karma, the test runner for JavaScript. This guide covers configuration, common commands, debugging techniques, and best practices to streamline your testing workflow.

Karma Configuration

Basic Configuration File (karma.conf.js)

The `karma.conf.js` file configures Karma's behavior. It specifies the files to be included in the test environment, testing framework, browsers to launch, and reporters to use.

```
module.exports = function(config) {
  config.set({
    frameworks: ['jasmine'],
    files: [
      'src/**/*.js',
      'test/**/*.spec.js'
    ],
    reporters: ['progress'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    browsers: ['Chrome'],
    autoWatch: true,
    singleRun: false
  });
};
```

Key Configuration Options

`frameworks`	An array of testing frameworks to use (e.g., 'jasmine', 'mocha', 'qunit').
`files`	An array of file patterns to load. Order matters; dependencies should be listed first.
`exclude`	An array of file patterns to exclude from loading.
`reporters`	An array of reporters to use (e.g., 'progress', 'dots', 'coverage').
`port`	The port Karma will listen on.
`browsers`	An array of browsers to launch for testing (e.g., 'Chrome', 'Firefox', 'Safari').
`autoWatch`	If true, Karma will watch files for changes and rerun tests automatically.
`singleRun`	If true, Karma will run tests once and exit.

Preprocessors

Preprocessors apply transformations to files before they are served to the browser. Common use cases include transpiling code (e.g., Babel for ES6) and generating coverage reports.

```
preprocessors: {
  'src/**/*.js': ['babel', 'coverage']
},
```

Running Karma Tests

Basic Commands

`karma start`	Starts the Karma test runner using the configuration file (karma.conf.js).
`karma run`	Triggers a test run without restarting the Karma server. Requires the server to be already running.
`karma init`	Helps create a karma.conf.js file in the current directory.

Command-Line Options

You can override configuration options from the command line using `--`. For example, to run tests in Firefox, use `karma start --browsers Firefox`.

`--single-run` : Override the singleRun setting in the config file

`--browsers` : Override the browsers setting in the config file

`--port` : Override the port setting in the config file

Example Commands

Run tests in Chrome once and exit:
`karma start --single-run --browsers Chrome`

Run tests and keep watching for changes:
`karma start`

Debugging Karma Tests

Debugging Techniques

Karma provides several ways to debug your tests, including using browser developer tools and the `browserConsoleLog` configuration option.

Using Browser Developer Tools

1. **Open the browser's developer tools**	Launch your tests using Karma, then open the developer tools in the browser (e.g., Chrome DevTools, Firefox Developer Tools).
2. **Set breakpoints**	Insert `debugger;` statements in your code or set breakpoints in the developer tools.
3. **Inspect variables**	Use the console or debugger to inspect variables and step through your code.

browserConsoleLog

The `browserConsoleLog` configuration option allows you to log messages from the browser console to the Karma console.

```
config.set({
  browserConsoleLogOptions: {
    level: 'debug',
    format: '%b %T: %m',
    terminal: true
  }
});
```

Advanced Karma Features

Custom Launchers

You can configure custom browser launchers to run tests in specific environments, such as headless Chrome or custom browser configurations.

```
customLaunchers: {
  ChromeHeadlessCI: {
    base: 'ChromeHeadless',
    flags: ['--no-sandbox']
  }
},
browsers: ['ChromeHeadlessCI']
```

Plugins

Karma supports a wide range of plugins to extend its functionality, including reporters, preprocessors, and frameworks. Install plugins using npm and configure them in your `karma.conf.js` file.

Reporters

`progress`	Displays a progress bar and test results in the console.
`dots`	Displays test results using dots in the console.
`coverage`	Generates code coverage reports.
`junit`	Generates JUnit-style XML reports.