**JSHint – A Javascript Static Analysis Tool**

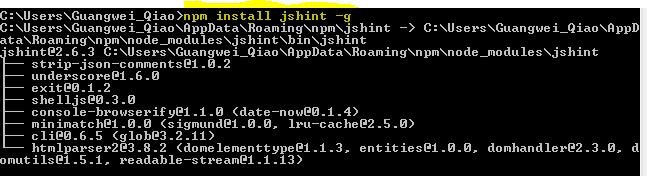
# JSHint Overview

JSHint is a very useful javascript static analysis tool which can help detect errors and potential problems in javascript code. The code checking standard of JSHint is very flexible and configurable.

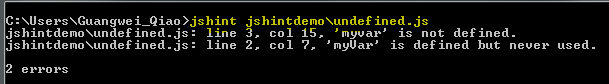
# JSHint Usage

## Basic Usage

The easiest way to use JSHint is to install it as a Node program. To do so, simply run the following command in your terminal (flag -g installs JSHint globally on your system, omit it if you want to install JSHint in the current working directory):



After you've done that you should be able to use the jshint program. The simplest use case would be linting a single file or all JavaScript files in a directory:



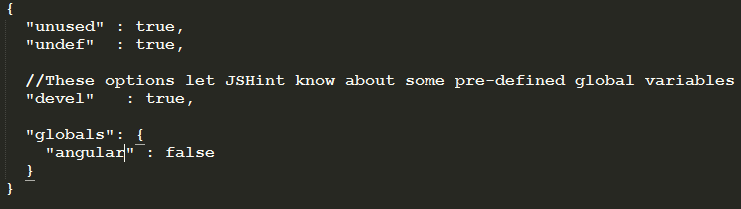
## Configuration

JSHint comes with a default set of warnings but it was designed to be very configurable. You have three methods to configure your copy of JSHint:

* Specify configuration file manually via the --config flag.
* Put you config into your projects package.json file under jshintconfig property.(For grunt project)
* Use a special file .jshintsrc

The third method is recommended, using .jshintsrc, JSHint will start looking for this file in the same folder as the files to be linted. If not found, it will move one level up the directory tree all the way up to the file system root. You can make use of this character, place your file into the project root directory and, as long as you run JSHint from anywhere within your project directory tree, the same configuration file will be used.

Configuration file is a simple JSON file that specifies which JSHint options to turn on or off. For example, the following file will enable warnings about undefined and unused variables and tell JSHint about a global variable named “angular” as read-only.

.

Each checking option can be added or deleted. [This page](http://jshint.com/docs/options/) contains a list of all options supported by JSHint.

# Useful Plugins for Text Editor or IDE

JSHint provides many amazing plugins for both text Editors and IDES. [This page](http://jshint.com/install/) contains a list of all plugins supported by JSHint. We take plugins(sublime-JSHint and JSHint-Eclipse) for example.

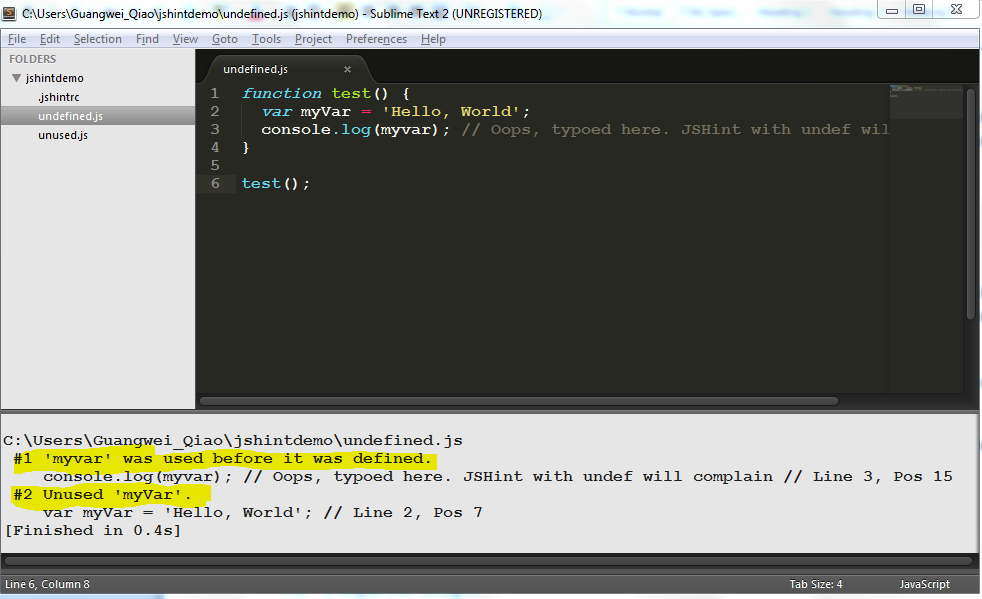
## Plugin for Sublime

Sublime is very convenient text editor. JSHint also provide some plugins for Sublime.

1. Install JSHint via npm, [see section 2.1](#_Basic_Usage).
2. Install Sublime 2(free edition).
3. Please use [Package Control](https://sublime.wbond.net/installation) to install the linter plugin. This will ensure that the plugin will be updated when new versions are available. To install via Package Control, do the following:

* Within Sublime Text, bring up the [Command Palette](http://docs.sublimetext.info/en/sublime-text-3/extensibility/command_palette.html) and type install. Among the commands you should see Package Control: Install Package. If that command is not highlighted, use the keyboard or mouse to select it. There will be a pause of a few seconds while Package Control fetches the list of available plugins.
* When the plugin list appears, type jshint. Among the entries you should see Sublime-jshint. If that entry is not highlighted, use the keyboard or mouse to select it.

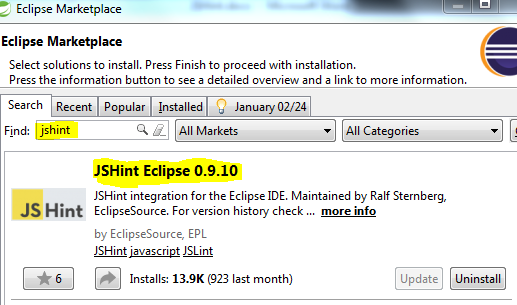
A snapshot shows the usage of sublime-JSHint as below:



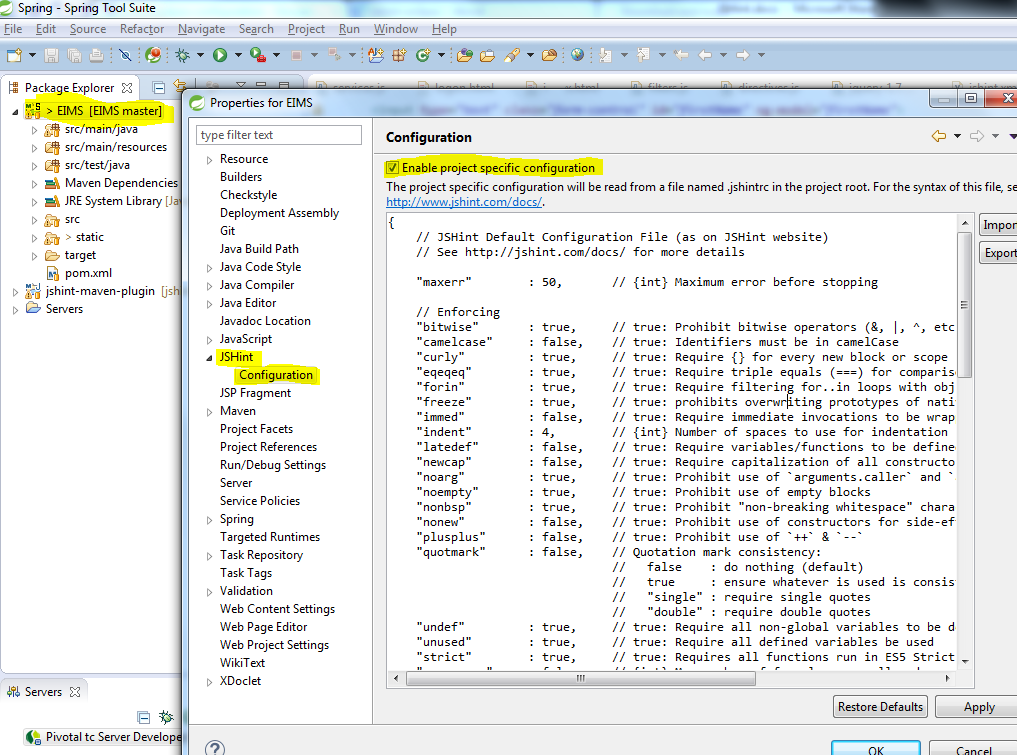
## Plugin for Eclipse

There is also a JSHint plugin for Eclipse or STS which is our frequently-used IDE. We can install the plugin through steps as below:

1. Launch Eclipse or STS.
2. Click Help-> Eclipse Marketplace… -> search “jshint”.
3. Install JSHint Eclipse 0.9.0 plugin as below snapshot.



After installation, you should configure the checking options you want as below,

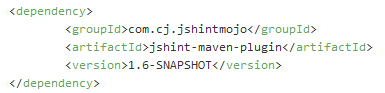


You can get the corresponding errors or warnings in “Problems” console of Eclipse.

# How to Integrate JSHint with Maven

For Appliance project, we prefer Maven as our building tool. So how to integrate JSHint with Maven is a task we are facing.

[jshint-mojo](https://github.com/cjdev/jshint-mojo) is a recommended plugin for Maven. To using jshint-mojo, some important parts in the pom.xml are as blow, (you can also refer the pom.xml in <https://github.com/guangweiqiao/EIMS>)





Note that, the current release version of jshint-mojo in [MVN Repository](http://mvnrepository.com/artifact/com.cj.jshintmojo/jshint-maven-plugin)  is [1.3.0](http://mvnrepository.com/artifact/com.cj.jshintmojo/jshint-maven-plugin) which is older than we need. So we’d better build a latest maven dependency from source code in github. And then you can get the latest version: 1.6-SNAPSHOT in your local maven repository.

It is not a good method to build jshint-mojo from source code every time for a new building environment, the better choice is building once and then you can get the ‘jshint-maven-plugin-1.6-SNAPSHOT.jar’ and ‘jshint-maven-plugin-1.6-SNAPSHOT.pom’ in your local maven repository. For other new building environment, we can use maven command to install plugin as below,

mvn install:install-file -Dfile=jshint-maven-plugin-1.6-SNAPSHOT.jar -DpomFile=jshint-maven-plugin-1.6-SNAPSHOT.pom

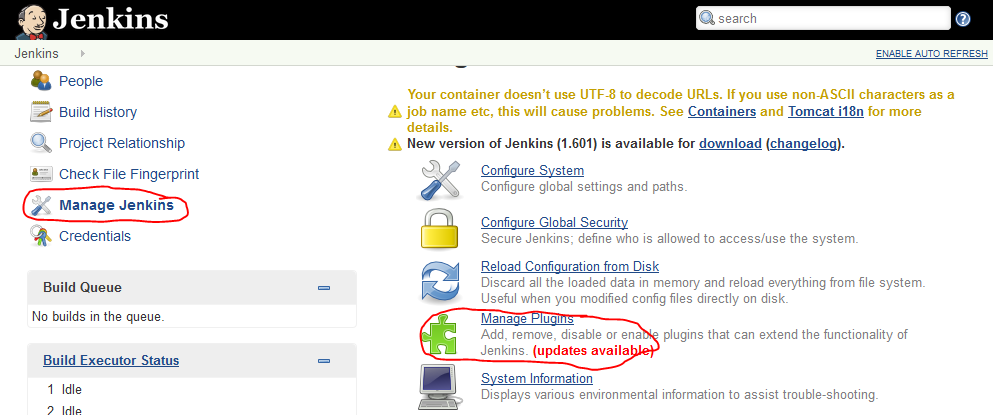
# How to Integrate JSHint with Jenkins

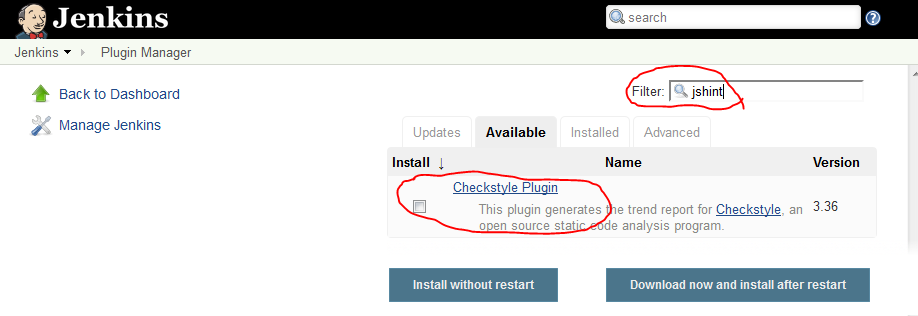
You know, we prefer Jenkins as our CI server, so can Jenkins integrate with JSHint seamlessly? The short answer is yes. Let’s start it.

I will ignore the process to install Jenkins. To be honest, there are many approaches to install a Jenkins in your local environment. Just for experiment, I merely deploy a Jenkins.war to a tomcat server in Windows system.

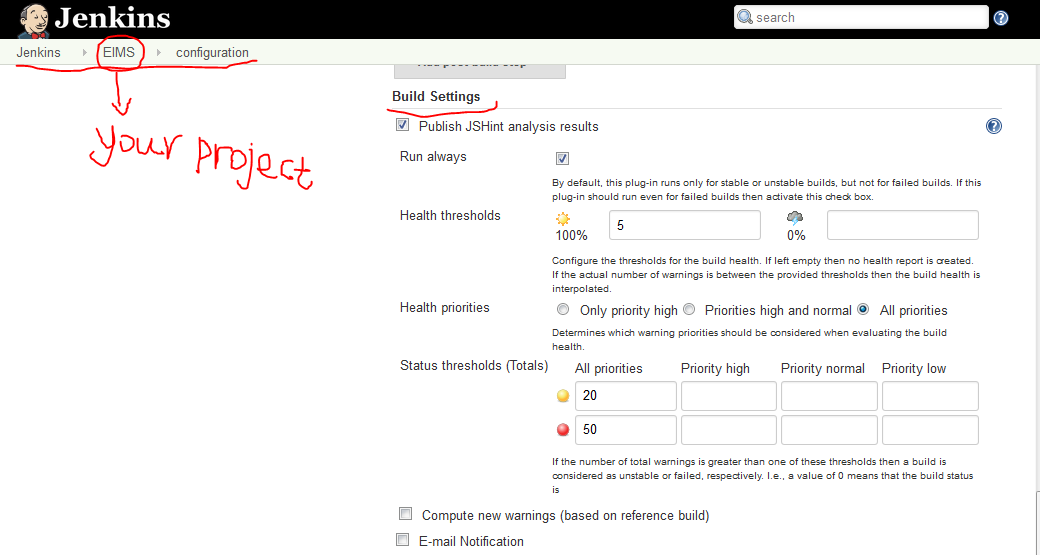
1. Install JSHint plugin into Jenkins.

Currently, the recommended JSHint plugin for Jenkins is ‘[Checkstyle Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Checkstyle+Plugin)’. Access you Jenkins Server, and click ‘Manage Jenkins’ -> ‘Manage Plugins’-> ‘Available’, and install the plugin.





1. After installation, you can configure your own project with JSHint settings.



1. After the customized settings, you can build your project.

