

VSCode Customization

User Settings

- Open “Preferences > Settings”
- Copy the followings and save

```
{
  "dart.previewFlutterUiGuides": true,
  "editor.wordWrap": "on",
  "editor.renderWhitespace": "boundary",
  "editor.tabSize": 2,
  "editor.insertSpaces": true,
  "editor.detectIndentation": false,
  "editor.suggestSelection": "first",
  "eslint.validate": [
    "javascript",
    "javascriptreact",
    "vue"
  ],
  "files.autoSave": "onFocusChange",
  "files.eol": "\n",
  "files.insertFinalNewline": true,
  "window.zoomLevel": 1,
  "workbench.startupEditor": "newUntitledFile"
}
```

ESLint

Reference: <https://code.visualstudio.com/docs/editor/extension-gallery>

- install ESLint with VS Code extension

See more details at [ESLint](#)

Common Extensions

Reference: <https://code.visualstudio.com/docs/editor/extension-gallery>

Name	Backend	Frontend	Flutter	Design	Optional	Remarks
Auto Close Tag	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Auto Rename Tag	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
C#	<input type="checkbox"/>	<input type="checkbox"/>				
Dart	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>		

Name	Backend	Frontend	Flutter	Design	Optional	Remarks
[Deprecated] Debugger for Chrome						
Docker	<input type="checkbox"/>					
DotENV	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Draw.io Integration	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
Error Lens					<input type="checkbox"/>	
ESLint	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		
Fig	<input type="checkbox"/>				<input type="checkbox"/>	You need to install Fig , and then will auto install the vscode extension. Currently only supports Mac, with cross-platform support coming soon.
Flutter	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>		
Gulp Tasks						Pls change to use Tasks Panel.
IntelliSense for CSS class names in HTML	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		
Kotlin Language			<input type="checkbox"/>			
Material Icon Theme (File Icon Theme)					<input type="checkbox"/>	
MongoDB for VS Code	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			Except for backend colleagues, please only use it for search purposes as much as possible.
npm	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		
npm Intellisense	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		
Path Intellisense	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
SQL Database Projects	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			Just for MSSQL Databases. Set it once and use it in different projects. The disadvantage is that switching tabs will execute sql again.
SQLTools	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			Need to setup in each projects. Except for backend colleagues, please only use it for search purposes as much as possible.
SQLTools Microsoft SQL Server/Azure	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			As SQLTools' driver.
Tasks Panel	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
tldraw					<input type="checkbox"/>	
Vetur	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		

Name	Backend	Frontend	Flutter	Design	Optional	Remarks
vscode-flutter-i18n-json			☐			
vscode-icons					☐	
Vue 3 Snippets	☐	☐		☐		A Vue.js 3 And Vue.js 2 Code Snippets Extension
webpack	☐	☐				


file icon setting:

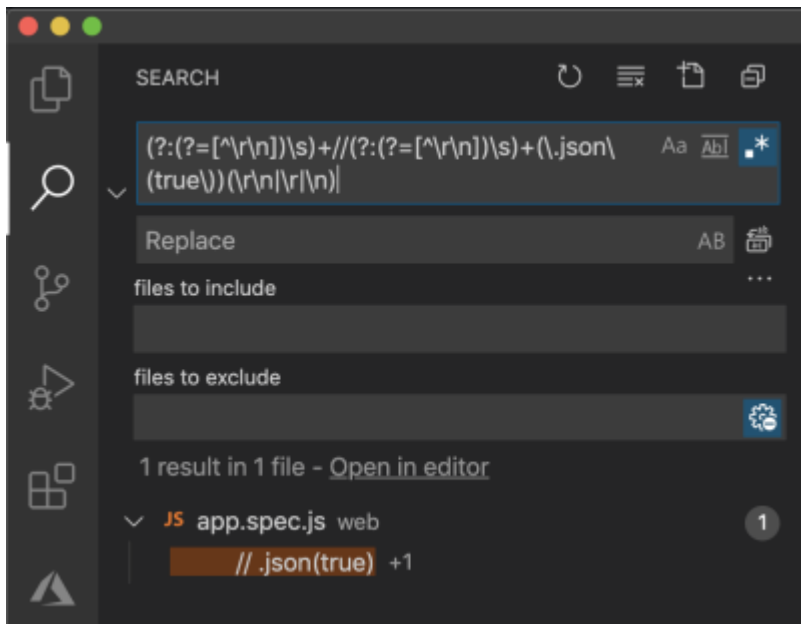


Use VS Code Regular Expression to search and replace

reference:

<https://docs.microsoft.com/en-us/visualstudio/ide/using-regular-expressions-in-visual-studio?view=vs-2019>

 VS Code allows you to search test or search by Regular Expression



whitespace: (?:(?=[^\r\n])\s)
 one or more whitespace: (?:(?=[^\r\n])\s)+
 new line character: (\r\n|\r|\n)
 optional: ?

- Combined Examples

`(?:(?=[^\r\n])\s)+/(?:(?=[^\r\n])\s)+(\.json\ (true\))(\r\n|\r|\n)`

```
will match the following, including the new line character at the end
(invisible)
// .json(true)
```

- using optional

```
(?:((?=[^\r\n])\s)+(/))?(?:((?=[^\r\n])\s)+(\.json\(\true\))(\r\n|\r|\n)
```

will match the following two by using optional // by using (/)?, including the new line character at the end (invisible)

```
// .json(true)
.json(true)
```

Use VS Code to Debug

Reference: <https://github.com/Microsoft/vscode-recipes/tree/master/nodemon>

We could use VS code to debug BOTH node.js backend and frontend codes at the same time. We use "congress.system" as an example, so change your startup script name, port, etc. accordingly.

Prerequisite

- install nodemon (to hot reload backend scripts for node.js)

```
$ npm install --save-dev nodemon
```

- modify package.json to run dev mode with nodemon

```
...
"scripts": {
  ...
  "dev": "cross-env NODE_ENV=development nodemon --inspect
congress.system.js", // change your startup script name here
},
...
```

Launch.json

Config the launch.json with different configurations:

```
{
  // Use IntelliSense to learn about possible Node.js debug attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit:
  https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
```

```
"configurations": [  
  {  
    "type": "node",  
    "request": "launch",  
    "name": "Launch Program",  
    "program": "${workspaceRoot}/congress.system.js", // change your  
startup script name here  
    "cwd": "${workspaceRoot}"  
  },  
  {  
    "type": "node",  
    "request": "attach",  
    "name": "Attach to Process",  
    "port": 5858  
  },  
  {  
    "type": "node",  
    "request": "attach",  
    "name": "Node: Nodemon",  
    "processId": "${command:PickProcess}",  
    "restart": true,  
    "protocol": "inspector",  
  },  
  {  
    "type": "chrome",  
    "request": "launch",  
    "name": "vuejs: chrome",  
    "url": "https://0.0.0.0:8080", // change your IP, port here  
    "webRoot": "${workspaceFolder}/public", // change your folder here  
    "breakOnLoad": true,  
    "sourceMapPathOverrides": {  
      "webpack:///public/*": "${webRoot}/*"  
    }  
  }  
]  
}
```

Explanations

- The first 2 configurations are standard node.js debug configurations created by VS Code.
- The 3rd configuration is added to "attach" the debugger to running node process with nodemon
- The 4th configuration is to open the Chrome browser with connecting to the debugger

Debugging

Follow these steps to run the debugger

- Start the node backend by running the command in VS Code terminal: "npm run dev"
- Go to VS Code debug pane and run "Node: Nodemon" from the dropdown

- when prompt, select the running node process corresponding to the “npm run dev”
- Go to VS Code debug pane and run “vuejs: chrome” from the dropdown
 - it will open Chrome

You could now debug both backend and frontend in VS Code.

- when you edit any backend file, nodemon will auto restart the node process

Using VS Code to run Mocha

You could run the Mocha test inside VS code so that you could even put a breakpoint when running test.

e.g. you have a test script in package.json

```
...
"scripts": {
  "test:lib": "NODE_ENV=test mocha --exit 'lib/test.setup.js'
'lib/**/*.spec.js'",
  "test:models": "NODE_ENV=test mocha --exit 'models/test.setup.js'
'models/**/*.spec.js'",
}
```

You could add a new configuration to your VS code “Launch.json” as follows. The most important part is the “args” list where you put your original arguments there.

```
...
"configurations": [
  ...
  {
    "type": "node",
    "request": "launch",
    "name": "test:lib",
    "env": {"NODE_ENV": "test"},
    "program": "${workspaceFolder}/node_modules/mocha/bin/_mocha",
    "args": [
      "--timeout",
      "999999",
      "--colors",
      "--exit",
      "${workspaceFolder}/lib/test.setup.js",
      "${workspaceFolder}/lib/**/*.spec.js"
    ],
    "internalConsoleOptions": "openOnSessionStart"
  },
  {
    "type": "node",
    "request": "launch",
```

```
"name": "test:models",
"env": {"NODE_ENV": "test"},
"program": "${workspaceFolder}/node_modules/mocha/bin/_mocha",
"args": [
  "--timeout",
  "999999",
  "--colors",
  "--exit",
  "${workspaceFolder}/models/test.setup.js",
  "${workspaceFolder}/models/**/*.spec.js"
],
"internalConsoleOptions": "openOnSessionStart"
},
]
```



we may change "user interface" from "tdd" to "bdd"

Using VS Code to run mochapack (for vue)

- Test components in the '/public/upload' etc. modules

File /webpack/vue.test.config.js

```
'use strict'

const merge = require('webpack-merge')
const nodeExternals = require('webpack-node-externals')
const vueBaseConfig = require('./vue.base.config.js')

module.exports = merge(vueBaseConfig, {
  output: {
    devtoolModuleFilenameTemplate: '[absolute-resource-path]',
    devtoolFallbackModuleFilenameTemplate: '[absolute-resource-path]?[hash]'
  },
  devtool: 'inline-cheap-module-source-map',
  externals: [nodeExternals()]
})
```

File /public/test.setup.js

```
'use strict'

require('jsdom-global')()
const chai = require('chai')
const sinonChai = require('sinon-chai')

chai.use(sinonChai)
```

File package.json

```
...
"scripts": {
  "test:public": "cross-env NODE_ENV=test mochapack --colors --watch --
webpack-config webpack/vue.test.config.js --require 'public/test.setup.js'
'public/**/*.spec.js'",
}
```

add configuration to vscode launch.json file:

```
...
"configurations": [
  ...
  {
    "type": "node",
    "request": "launch",
    "name": "test:public",
    "env": {"NODE_ENV": "test"},
    "program": "${workspaceFolder}/node_modules/mochapack/bin/mochapack",
    "args": [
      "--colors",
      "--watch",
      "--webpack-config",
      "${workspaceFolder}/webpack/vue.test.config.js",
      "--require",
      "${workspaceFolder}/public/test.setup.js",
      "${workspaceFolder}/public/**/*.spec.js",
    ]
  }
]
```

Using VS Code to run Cucumber

Install cucumber-js and add features and steps

- npm install --save-dev cucumber
- add /features/xxx.feature
- add /features/step_definitions/xxx.js

```
...
"configurations": [
  ...
  {
    "type": "node",
    "request": "launch",
    "name": "test-bdd",
    "env": {"NODE_ENV": "test"},
  }
]
```

```
"program": "${workspaceFolder}/node_modules/.bin/cucumber-js",
"args": [
  "${workspaceFolder}/features/**/*.feature",
  "-r",
  "${workspaceFolder}/features/step_definitions/**/*.js"
],
"console": "integratedTerminal"
}
]
```

From:

<https://wiki.questwork.com/dokuwiki/> - Questwork's Wiki

Permanent link:

<https://wiki.questwork.com/dokuwiki/doku.php?id=development:tools:vscode:start&rev=1648624034>

Last update: **2022/03/30 15:07**

