

Your First VS Code Extension

Andrei Voronin

About

- Ecommerce CMS
- Frontend
- Storefront &
Admin panel UX/UI



Accelerate development



Accelerate development

- Command palette
- Custom shortcuts/snippets
- Extensions
- Custom extensions



Visual Studio Code

The screenshot shows the Visual Studio Code interface. On the left, the Extensions Marketplace is open, displaying a list of popular extensions:

- Python
- GitLens
- C/C++
- ESLint
- Debugger for Chrome
- Language Support
- vscode-icons
- Vetur

On the right, a Gatsby GraphQL application is being developed. The code editor shows `blog-post.js` and `index.js`. The `index.js` file contains the following code:

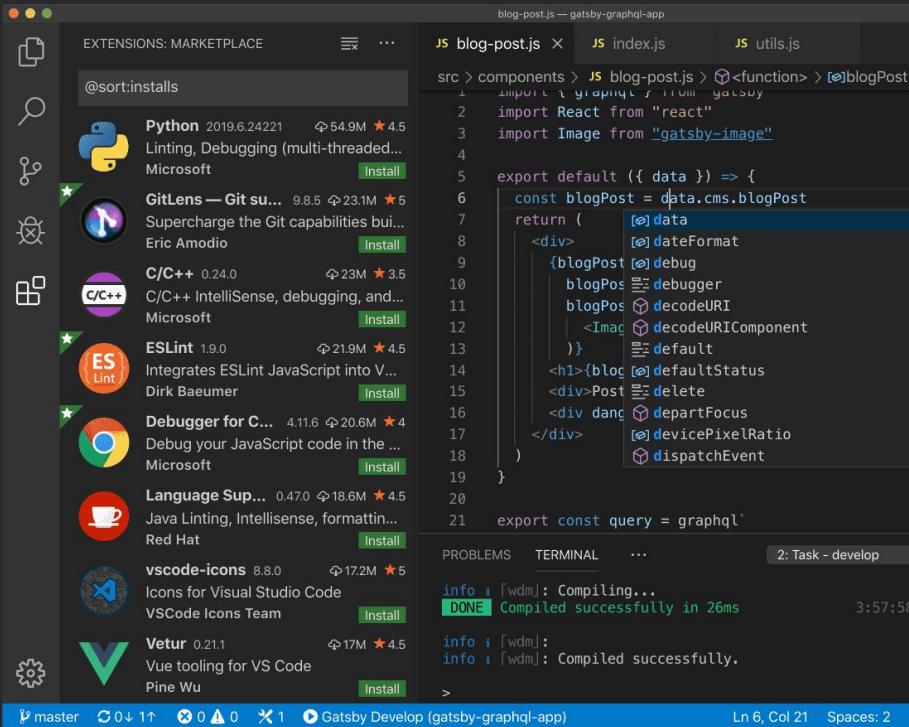
```
src > components > JS blog-post.js > index.js > utils.js
1 import React from "react"
2 import Image from "gatsby-image"
3
4 export default ({ data }) => {
5   const blogPost = data.cms.blogPost
6   return (
7     <div>
8       {blogPost.debug}
9       <img alt={decodeURI(blogPost.image.default)} />
10      <h1>{blogPost.defaultStatus}</h1>
11      <div>Post</div>
12      <div dang=>departFocus
13        <div>devicePixelRatio
14        <div>dispatchEvent
15      </div>
16    </div>
17  )
18
19 }
20
21 export const query = graphql`
```

The terminal at the bottom shows the build process:

```
info i [wdm]: Compiling...
DONE Compiled successfully in 26ms
info i [wdm]:
info i [wdm]: Compiled successfully.
```

Visual Studio Code

- Fast & free
- Built-in Git + GitLens
- Actively developing

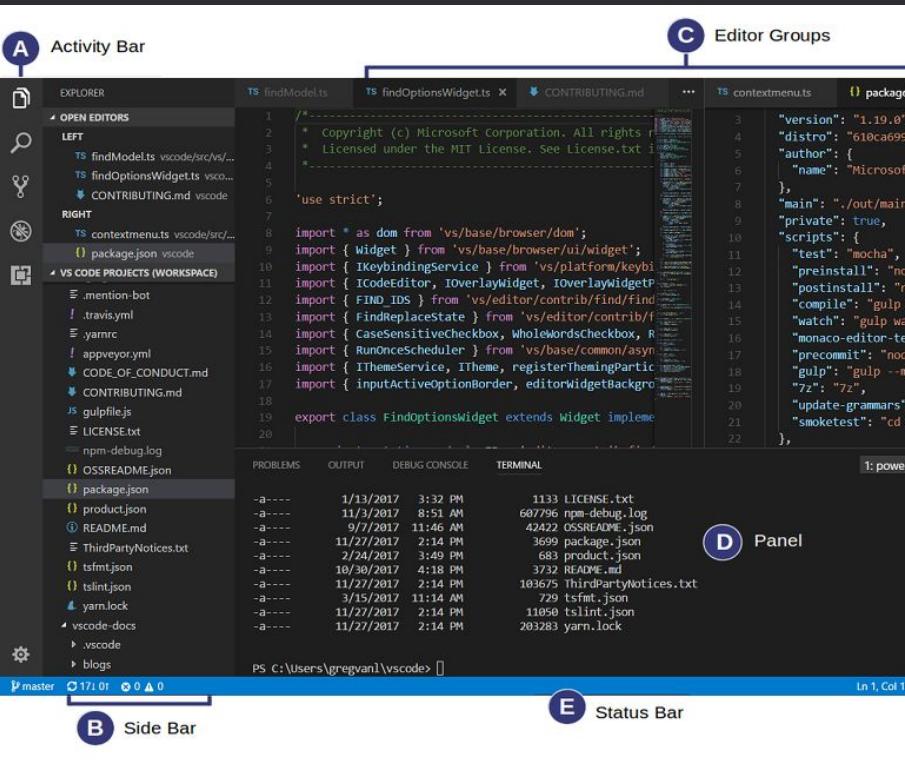


The screenshot shows the Visual Studio Code interface with the following components:

- Extensions Marketplace:** Opened in the sidebar, showing popular extensions like Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support, and vscode-icons.
- Code Editor:** Displays a file named "blog-post.js" from a "gatsby-graphql-app" repository. The code uses JSX and imports from "react" and "gatsby-image". A tooltip highlights the use of the "data" prop.
- Terminal:** Shows the command "Task - develop" and output indicating a successful build: "Compiled successfully in 26ms".
- Status Bar:** Shows the current branch ("master"), file count (0), error count (0), task count (1), and the active workspace ("Gatsby Develop (gatsby-graphql-app)"). It also indicates the current line (Ln 6), column (Col 21), and total spaces (Spaces: 2).

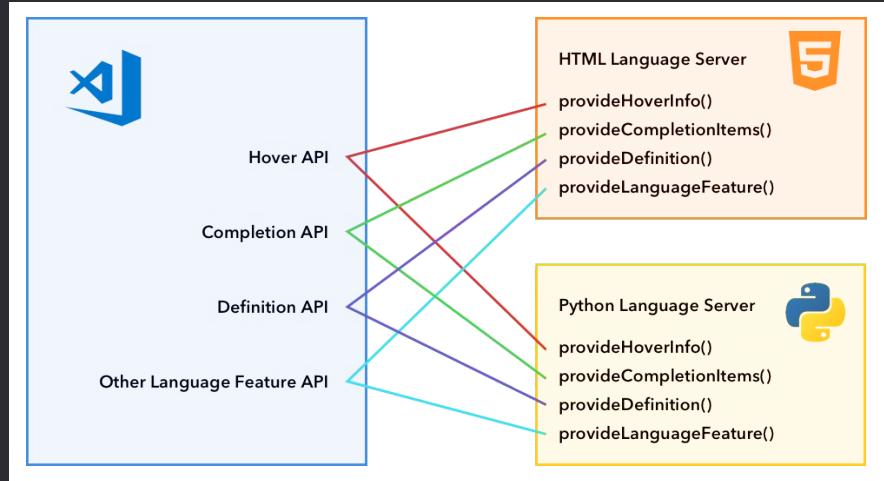
Built with extensibility in mind

- Fully customizable
- Core features as extensions
- Like browser extensions



Capabilities

- Syntax highlighting
- Hover info & completion
- Jump to definition
- Error checking
- Shortcut & context menu
- Notification & status bar info
- Activity bar view



Hello world

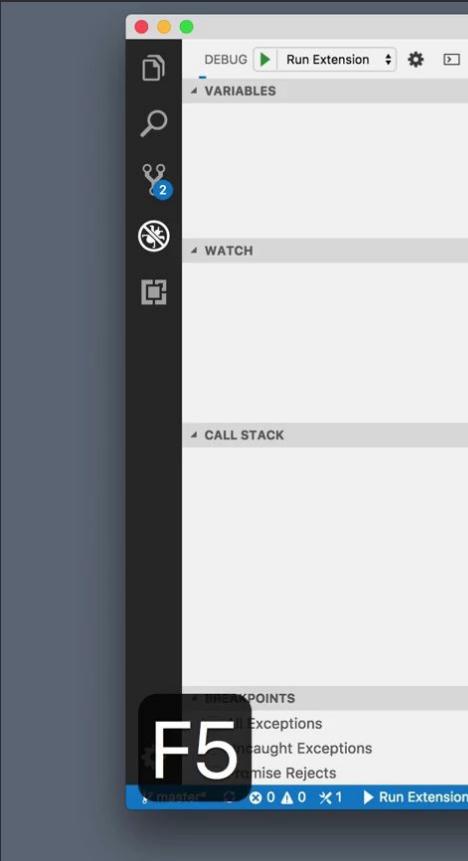
Dev environment

- `yarn global add yo generator-code`
 - `yo code`
 - New Extension (TypeScript)
 - `code ./helloworld`
- ```
? What type of extension do you want to create?
 New Extension (TypeScript)
? What's the name of your extension?
 HelloWorld
 ### Press <Enter> to choose default
 for all options below ###

? What's the identifier of your extension?
 helloworld
? What's the description of your extension?
 LEAVE BLANK
? Initialize a git repository?
 Yes
? Which package manager to use?
 yarn
```

# Run

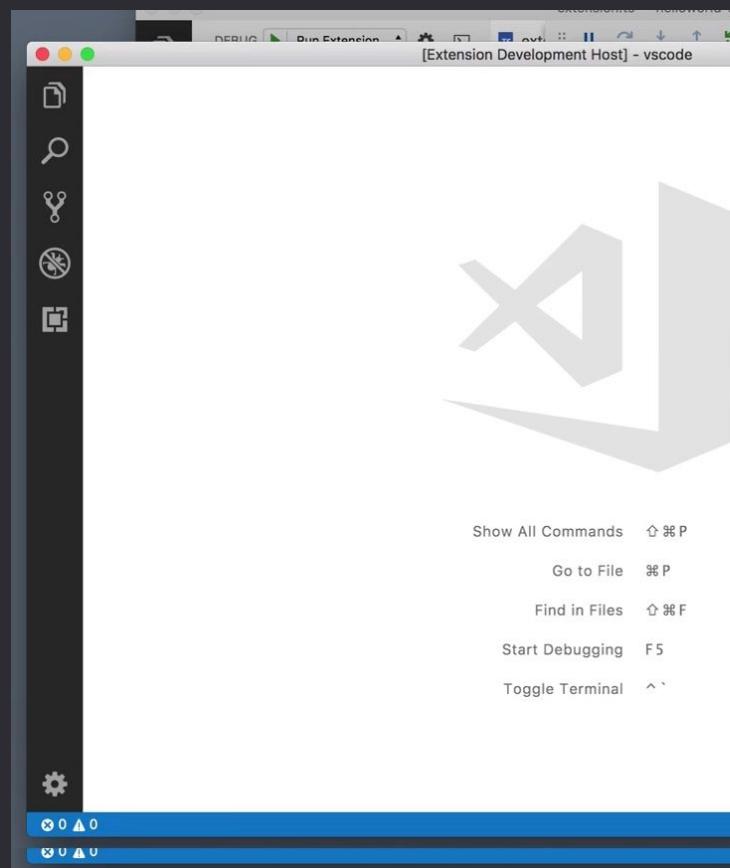
1. Start debugging F5
2. Extension Development Host window
3. Command Palette 
4. Run the Hello World
5. Notification showing up



```
extension.ts — helloworld-sample
1 // The module 'vscode' contains the VS Code API
2 // Import the module and reference it with the $1 identifier
3 import * as vscode from 'vscode';
4
5 // this method is called when your extension
6 // is activated the first time
7 export function activate(context: vscode.ExtensionContext) {
8 // Use the console to output diagnostic information
9 // This line of code will only be executed in the context of the extension's activation
10 console.log('Congratulations, your extension has been activated');
11
12 // The command has been defined in the package.json file
13
14 // Now provide the implementation of the command
15 // The commandId parameter must match the command declaration in package.json
16 let disposable = vscode.commands.registerCommand(
17 'extension.helloWorld',
18 // The code you place here will be executed when the command is run
19 () => {
20 // Display a message box to the user
21 vscode.window.showInformationMessage(`Hello ${vscode.env.user.name}!`);
22
23 context.subscriptions.push(disposable);
24 }
25
26 // this method is called when your extension
27 export function deactivate() {}
28 }
```

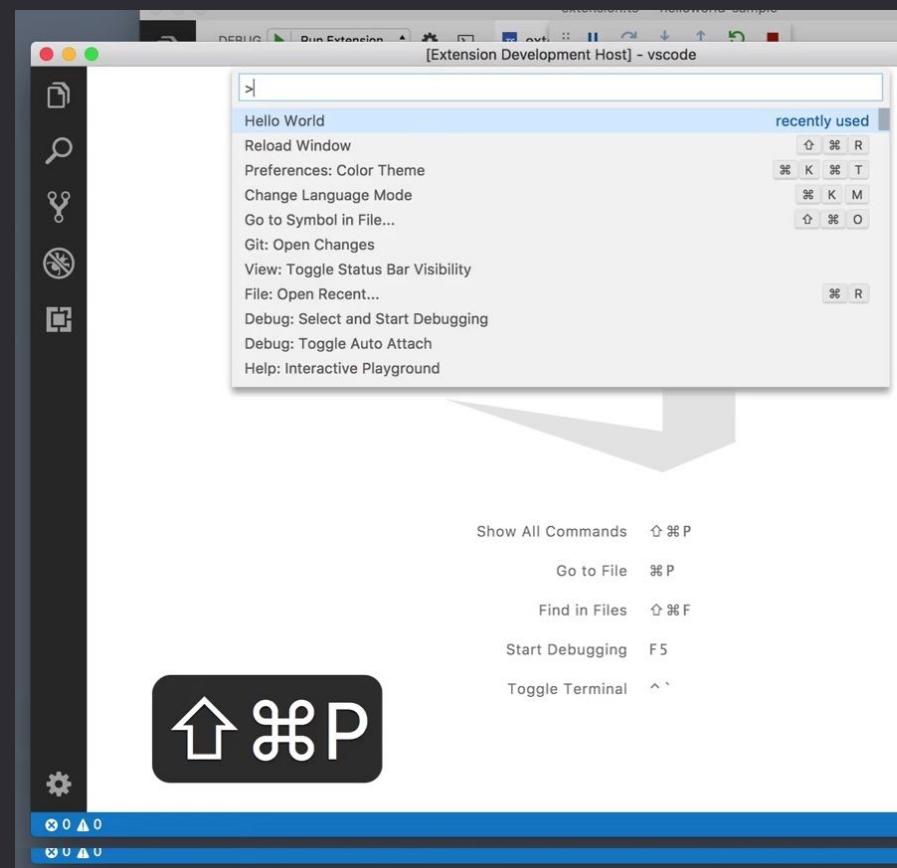
# Run

1. Start debugging F5
2. Extension Development Host window
3. Command Palette  $\text{Shift} + \text{Command} + \text{P}$
4. Run the Hello World
5. Notification showing up



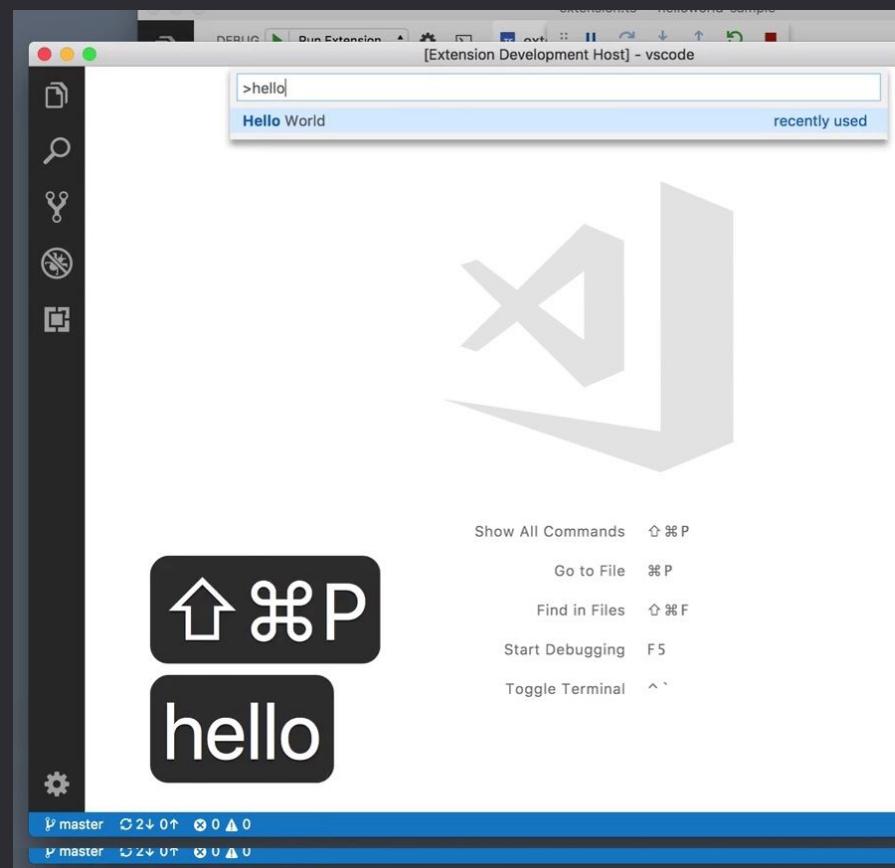
# Run

1. Start debugging F5
2. Extension Development Host window
3. Command Palette 
4. Run the Hello World
5. Notification showing up



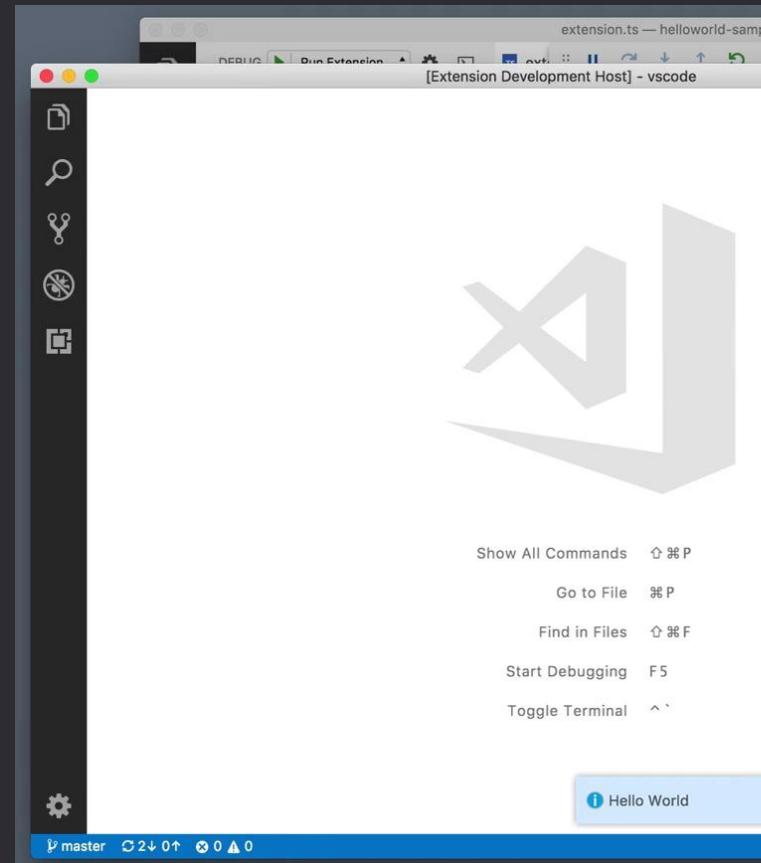
# Run

1. Start debugging F5
2. Extension Development Host window
3. Command Palette  $\text{\textuparrow}$   $\mathbin{\text{\texttilda}}$  P
4. Run the Hello World
5. Notification showing up



# Run

1. Start debugging F5
2. Extension Development Host window
3. Command Palette  $\text{Shift} + \text{Command} + \text{P}$
4. Run the Hello World
5. **Notification showing up**



# Structure

- Extension Manifest
  - package.json
- Extension Entry File
  - extensions.ts

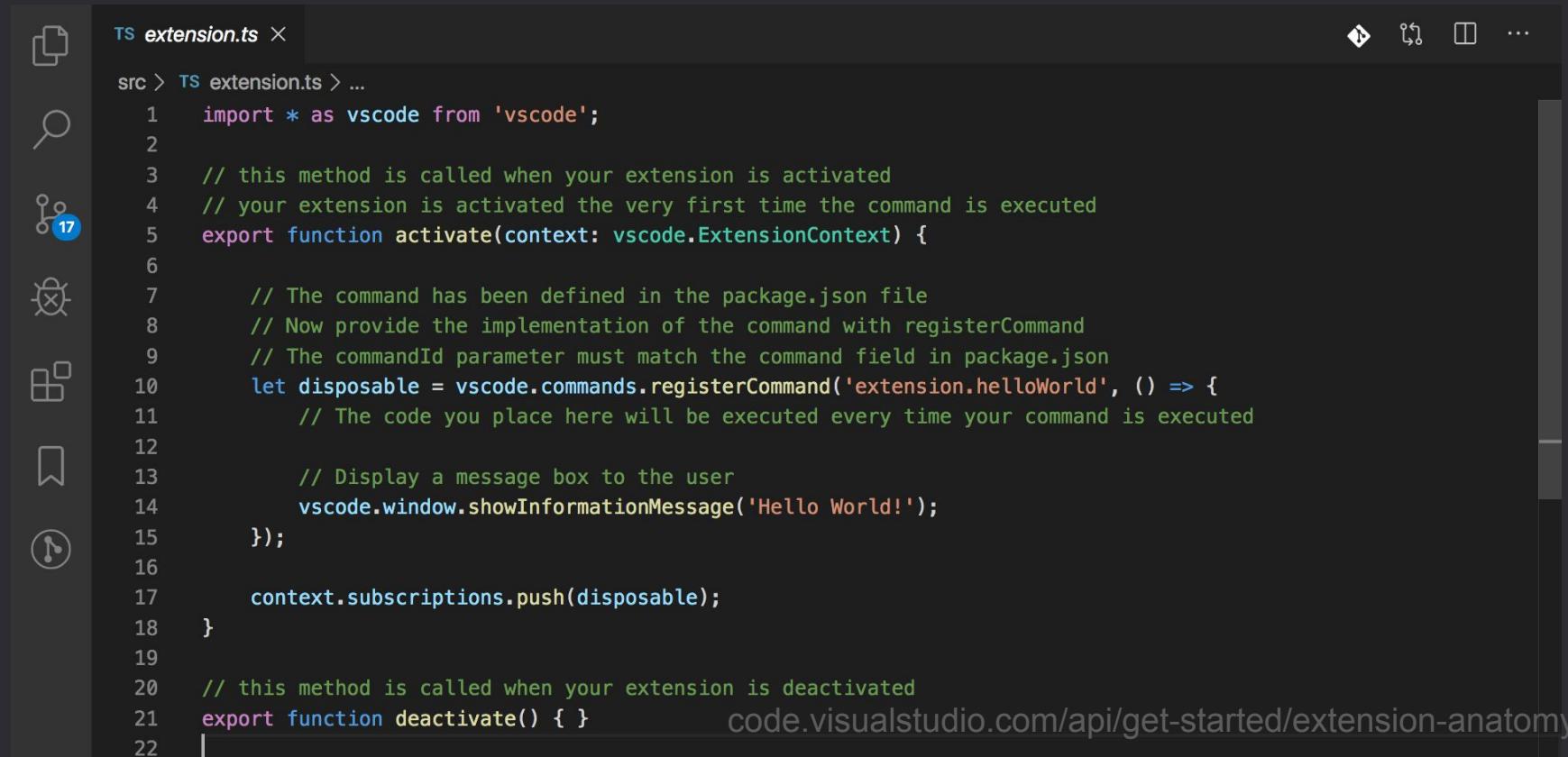
The screenshot shows the VS Code Explorer sidebar with the following file structure:

- HELLOWORLD
  - .vscode
    - extensions.json
    - launch.json
    - settings.json
    - tasks.json
  - node\_modules
  - out
    - test
    - extension.js
    - extension.js.map
  - src
    - test
    - extension.ts
  - .gitignore
  - .vscodeignore
  - CHANGELOG.md
  - package.json
  - README.md
  - tsconfig.json
  - tslint.json
  - vsc-extension-quickstart.md
  - yarn.lock

# Manifest package.json

```
{
 "name": "helloworld-sample",
 "publisher": "vscode-samples",
 "main": "./out/extension.js",
 "activationEvents": ["onCommand:extension.helloWorld"], // ["*"]
 "contributes": {
 "commands": [
 {
 "command": "extension.helloWorld",
 "title": "Hello World"
 }
]
 },
}
code.visualstudio.com/api/get-started/extension-anatomy
```

# Entry File extensions.ts



The screenshot shows the Visual Studio Code interface with the 'extension.ts' file open in the main editor area. The file contains TypeScript code for a VS Code extension. The code defines an 'activate' function that registers a command ('extension.helloWorld') which displays an information message ('Hello World!'). It also defines a 'deactivate' function. The code editor has syntax highlighting for TypeScript and includes line numbers. On the left, the Explorer sidebar shows a folder structure with 'src' expanded, containing 'extension.ts'. The status bar at the bottom right shows the URL 'code.visualstudio.com/api/get-started/extension-anatomy'.

```
TS extension.ts ×
src > TS extension.ts > ...
1 import * as vscode from 'vscode';
2
3 // this method is called when your extension is activated
4 // your extension is activated the very first time the command is executed
5 export function activate(context: vscode.ExtensionContext) {
6
7 // The command has been defined in the package.json file
8 // Now provide the implementation of the command with registerCommand
9 // The commandId parameter must match the command field in package.json
10 let disposable = vscode.commands.registerCommand('extension.helloWorld', () => {
11 // The code you place here will be executed every time your command is executed
12
13 // Display a message box to the user
14 vscode.window.showInformationMessage('Hello World!');
15 });
16
17 context.subscriptions.push(disposable);
18 }
19
20 // this method is called when your extension is deactivated
21 export function deactivate() { }
code.visualstudio.com/api/get-started/extension-anatomy
```

# Anatomy

1.

Registers the  
`onCommand`

Activation Event

`package.json`

2.

Uses the  
`contributes.commands`

Contribution Point

`package.json`

3.

Uses the  
commands.  
`registerCommand`

VS Code API

`extensions.ts`



# ГЕРОИ БОЛЬШОГО ЭКРАНА. ВСТРЕЧАЙТЕ IPHONE XS

Дисплей Super Retina в двух размерах, один из которых стал самым большим в истории iPhone. Самый мощный и умный процессор iPhone. И потрясающая двойная камера с функцией «Глубина». В iPhone XS воплощено всё, что вы любите в iPhone. На новом уровне.



iPhone XS

## ХИТЫ ПРОДАЖ / БЕСТСЕЛЛЕРЫ

[Хиты продаж](#) [Распродажа](#) [Самые популярные](#)


Sony - Platinum Wireless 7.1 Virtual Surround Sound...

11 365 ₽

10 365 ₽



Sony - DualShock 4 Wireless Controller for So...

2 591 ₽



Sony PlayStation 4 Dual-Shock 4 Controller, Green...

2 591 ₽



Sony PlayStation 4 Dual-Shock 4 Wireless Controll...

2 780 ₽



Sony PlayStation 4 Dual-Shock 4 Wireless Controll...

2 780 ₽

УМНАЯ КОЛОНКА,  
НОМЕРОДУправляйте своим умным домом  
с помощью голоса.APPLE  
WATCH  
NIKE+

Подключайтесь через спорт.

Сканировать



Simtech

Модули Администрирование Настройки Дизайн RU

Быстрое меню

Окт 21, 2019 — Ноя 21, 2019

### Панель инструментов

ПОСМОТРЕТЬ, КАК ЭТО РАБОТАЕТ

|                               |                 |
|-------------------------------|-----------------|
| Продажи                       | 0 ₽<br>0 ₽, 00% |
| Налоги                        | 0 ₽<br>0 ₽, 00% |
| Товары на витрине             | 263             |
| Нет в наличии                 | 10              |
| Зарегистрированные покупатели | 12              |
| Категории                     | 117             |
| Магазины                      | 1               |
| Веб-страницы                  | 10              |

### Статистика

Продажи

Последние заказы

Все Обработан На удержании Ожидает звонка Аннулирован Отложен Отклонен Неудача

Открыт Выполнен Возвращено

Здесь пока ничего нет

### Последние события

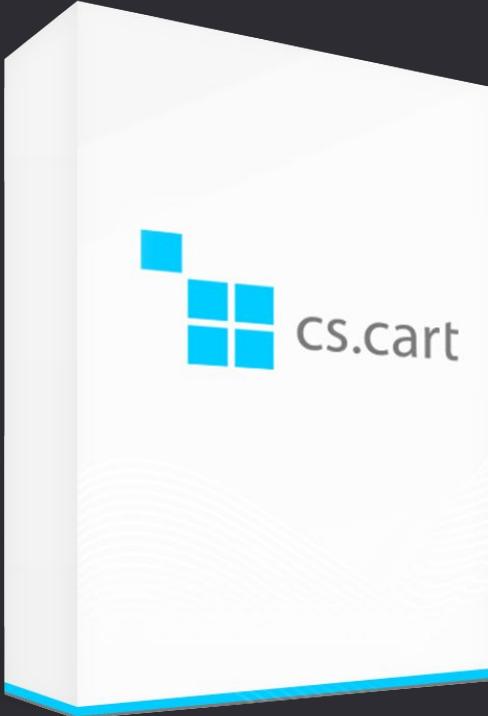
Показать все

Пользователи (Сессия): Администратор Главный; admin@example.com (#1)  
11/21/2019, 16:38

Пользователи (Сессия): Администратор Главный; admin@example.com (#1)  
11/21/2019, 16:38

Скачать

# CMS



# Framework

[github.com/andreivoronin/  
cscartcode](https://github.com/andreivoronin/cscartcode)

# Language variables

```
{\language_variables}
```

```
85 {dropdown content=$smarty.capture.tools_list}
86
87
88 {if $products}
89 {include file="buttons/save.tpl" but_name="dispatch[products.m_update]" but_role=}
90 {/if}
91 {/capture} msgctxt "Languages::text_select_fields2edit_note"
92 msgid "You are able to modify several items at once. Select the chec
93 {capture} k boxes corresponding to the fields you wish to edit, and click on M
94 odify Selected."
95 <p>{__("text_select_fields2edit_note")}</p>
96 {include file="views/products/components/products_select_fields.tpl"}
97
98 <div class="buttons-container">
99 {include file="buttons/save_cancel.tpl" but_text=__("modify_selected") but_name="disp
100 </div>
101 {/capture}
102 {include file="common/popupbox.tpl" id="select_fields_to_edit" text=__("select_fields_to_"
103
104 </form>
105
106 {/capture}
```

# extension.ts

```
1 import * as vscode from 'vscode';
2 import { LanguageVariableDefinitionProvider } from './LanguageVariableDefinitionProvider';
3
4 export function activate(context: vscode.ExtensionContext) {
5
6 let languageVariableDefinitionProvider = vscode.languages.registerDefinitionProvider(
7 {
8 scheme: 'file',
9 language: 'smarty',
10 }
11 , new LanguageVariableDefinitionProvider());
12
13 context.subscriptions.push(languageVariableDefinitionProvider);
14 }
15
16 export function deactivate() {}
17 |
```

# LanguageVariableDefinitionProvider.ts

- `provideDefinition(document: TextDocument, position: Position, token: CancellationToken): Promise<Definition | undefined>`
- `/_\\"[\w\.]+\"\)/`
- `line = document.lineAt(position)`
- `new Location(Uri.file(workspaceFolders + poFilePath), new Position(langVarLineNumber, 0))`

# LanguageVariableDefinitionProvider.ts

```
1 import fs = require('fs');
2 import {
3 CancellationToken,
4 TextDocument,
5 DefinitionProvider,
6 Definition,
7 Location,
8 Position,
9 workspace,
10 Uri,
11 TextLine
12 } from 'vscode';
13
14 const fsPromises = fs.promises;
15 const LANGVAR_MATCHER = /__\("[\w\.]+"\)/;
16 const START_LANGVAR_MATCHER = /__\("[\w\.]+\)/;
17 const START_LANGVAR_SYMBOL = '__("';
18 const END_LANGVAR_SYMBOL = '")';
19 const startLangvarSymbolLength = START_LANGVAR_SYMBOL.length;
20 const endLangvarSymbolLength = END_LANGVAR_SYMBOL.length;
21
22 let textBeforeLangVar = '';
23 let langVarLineNumber = 0;
24 let poFilePath = '/var/langs/en/core.po';
```

# LanguageVariableDefinitionProvider.ts

```
26 export class LanguageVariableDefinitionProvider implements DefinitionProvider {
27
28 public async provideDefinition(document: TextDocument, position: Position,
29 token: CancellationToken): Promise<Definition | undefined> {
30 if (!workspace.workspaceFolders) {
31 return;
32 }
33
34 const line = document.lineAt(position);
35 const tag = getLangVarFromLine(line);
36
37 if (!tag) {
38 return;
39 }
40
41 const workspaceFolders = workspace.workspaceFolders[0].uri.path;
42 const tagDirtyText = tag ? tag[0] : '';
43 const tagText = tagDirtyText.substring(startLangvarSymbolLength,
44 tagDirtyText.length - endLangvarSymbolLength);
45 const langvarPosition = getLangVarPosition(tag);
```

# LanguageVariableDefinitionProvider.ts

```
46
47 if (position.character < langvarPosition.start || position.character > langvarPosition.end){
48 return;
49 }
50
51 const fileText = await readFile(workspaceFolders, poFilePath) || '';
52 const langVarIndex = fileText.indexOf(tagText);
53
54 if (langVarIndex > 0) {
55 textBeforeLangVar = fileText.toString().substring(0, langVarIndex);
56 langVarLineNumber = textBeforeLangVar.split(/\r?\n/).length - 1;
57 }
58
59 return new Location(Uri.file(workspaceFolders + poFilePath), new Position(langVarLineNumber, 0));
60 }
61 }
```

```
63 async function readFile(workspaceFolders: string, poFilePath: string) {
64 try {
65 return fsPromises.readFile(workspaceFolders + poFilePath);
66 } catch (err) {
67 console.error('Error occurred while reading file!', err);
68 }
69 }
70
71 function getLangVarFromLine(line: TextLine) {
72 const tagStart = START_LANGVAR_MATCHER.exec(line.text);
73 const tagstartIndex = tagStart ? tagStart.index : 0;
74
75 return LANGVAR_MATCHER.exec(line.text.substring(tagstartIndex));
76 }
77
78 function getLangVarPosition(tag: RegExpExecArray | null) {
79 const start = tag ? tag.index + startLangvarSymbolLength + 1 : 0;
80 const tagLength = tag ? tag[0].length - endLangvarSymbolLength : 0;
81 const end = start + tagLength;
82
83 return { start, end, };
84 }
```

# Include

```
{include file="template.tpl"}
```

```
577 {if $id}
578 {** Product options section **}
579 <div class="cm-hide-save-button hidden" id="content_options">
580 {include file="views/products/components/products_update_options.tpl"}
581 </div>
582 {** /Product options section **}
583
584 {** Products files section **}
585 {if $settings.General.enable_edp == "Y"}
586 <div id="content_files" class="cm-hide-save-button hidden">
587 {hook name="products:content_files"}
588 {include file="views/products/components/products_update_files.tpl"}
589 {/hook}
590 </div>
591 {/if}
592 {** /Products files section **}
593
594 {** Subscribers section **}
595 <div id="content_subscribers" class="cm-hide-save-button hidden">
596 {include file="views/products/components/product_subscribers.tpl" product_id=$id}
597 </div>
598 {** /Subscribers section **}
599 {/if}
600
601 {/capture}
602 {include file="common/tabsbox.tpl" content=$smarty.capture.tabsbox group_name=$runtime.controller active_t
603
```

# extension.ts

```
1 import * as vscode from 'vscode';
2 import { IncludeLinkProvider } from './IncludeLinkProvider';
3
4 export function activate(context: vscode.ExtensionContext) {
5
6 let disposable = vscode.languages.registerDocumentLinkProvider(
7 {
8 scheme: 'file',
9 language: 'smarty',
10 }
11 , new IncludeLinkProvider());
12
13 context.subscriptions.push(disposable);
14 }
```

# IncludeLinkProvider.ts

- `provideDocumentLinks(document: TextDocument, token: CancellationToken): Promise<DocumentLink[]>`
- `/\{include/g, /file=\\"\\S*\\".tpl\\"/g`
- `line = document.lineAt(1);`
- `Position(line.lineNumber, fileMatchesLine.index + 6)`
- `new DocumentLink(new Range(start, end), url)`

# IncludeLinkProvider.ts

```
1 import * as path from 'path';
2 import fs = require('fs');
3 import * as vscode from 'vscode';
4 import {
5 CancellationToken,
6 DocumentLink,
7 DocumentLinkProvider,
8 TextDocument,
9 workspace
10 } from 'vscode';
11 import {
12 Position,
13 Range,
14 Uri
15 } from 'vscode';

16
17 const INCLUDE_MATCHER = /{include/g;
18 const FILE_MATCHER = /file=\"\S*\.\tpl\\"/g;
19

20 export class IncludeLinkProvider implements DocumentLinkProvider {
21 public async provideDocumentLinks(document: TextDocument, token: CancellationToken): Promise<DocumentLink[]> {
22 const urls = await findIncludeUrl(document);
23 return Promise.resolve(urls ? urls : []);
24 }
25 }
```

# IncludeLinkProvider.ts

```
27 async function findIncludeUrl(document: TextDocument): Promise<DocumentLink[] | undefined> {
28 const results: DocumentLink[] = [];
29 let line: vscode.TextLine;
30 let searchString: string = '';
31 let foundMultiline: Boolean = false;
32 let documentLink: DocumentLink | undefined;
33
34 for (let l = 0; l < document.lineCount; l++) {
35 line = document.lineAt(l);
36
37 if (!(INCLUDE_MATCHER.exec(line.text)) !== null && !line.text.includes('}')) {
38 searchString = line.text;
39 foundMultiline = true;
40
41 documentLink = await getLink(line);
42 if (documentLink) {
43 results.push(documentLink);
44 }
45
46 continue;
47
48 } else if (foundMultiline && !line.text.includes('}')) {
49 searchString += line.text;
50
51 documentLink = await getLink(line);
52 if (documentLink) {
53 results.push(documentLink);
```

# IncludeLinkProvider.ts

```
54 }
55
56 continue;
57 } else if (foundMultiline && line.text.includes('}')) {
58 searchString += line.text;
59 foundMultiline = false;
60
61 documentLink = await getLink(line);
62 if (documentLink) {
63 results.push(documentLink);
64 }
65
66 } else {
67 searchString = line.text;
68
69 documentLink = await getLink(line);
70 if (documentLink) {
71 results.push(documentLink);
72 }
73 }
74 }
75
76 return results;
77 }
78 }
```

# IncludeLinkProvider.ts

```
79 async function getLink(line: vscode.TextLine): Promise<DocumentLink | undefined> {
80 let fileMatchesLine: RegExpExecArray | null;
81 let link: string = '';
82 let url: Uri | undefined;
83 let start: Position = new Position(0, 0);
84 let end: Position = new Position(0, 0);
85
86 while ((fileMatchesLine = FILE_MATCHER.exec(line.text)) !== null) {
87 if (fileMatchesLine) {
88 link = fileMatchesLine[0].substring(6, fileMatchesLine[0].length - 1);
89 url = await getFileUrl(link);
90
91 start = new Position(line.lineNumber, fileMatchesLine.index + 6);
92 end = start.translate(0, link.length);
93
94 return new DocumentLink(new Range(start, end), url);
95 }
96 }
97 }
```

# IncludeLinkProvider.ts

```
50
99 async function getFileUrl(link: string): Promise<Uri | undefined> {
100 let validFilePath = await getValidFilePath(link);
101
102 if (!validFilePath) {
103 return;
104 }
105
106 return Uri.file(validFilePath);
107 }
108
109 async function getValidFilePath(link: string): Promise<string | undefined> {
110 let filePath = await getPath(link);
111
112 if (!filePath || !workspace.workspaceFolders) {
113 return;
114 }
115
116 return fs.existsSync(filePath) ?
117 filePath : [workspace.workspaceFolders[0].uri.path, 'design', 'backend', 'responsive', 'templates', link]
118 .filter(entry => entry.trim() !== '')
119 .join('/');
120 }
121
```

```
122 async function getFilePath(link: string): Promise<string | undefined> {
123 if (!vscode.window.activeTextEditor) {
124 await vscode.window.activeTextEditor;
125 }
126
127 if (!workspace.workspaceFolders) {
128 return;
129 }
130 if (!vscode.window.activeTextEditor) {
131 return;
132 }
133 let folderPath = '';
134
135 if (vscode.window.activeTextEditor) {
136 folderPath = workspace.asRelativePath(path.dirname(vscode.window.activeTextEditor.document.fileName));
137 }
138
139 let location = 'backend';
140 let theme = '';
141
142 if (folderPath.split('/')[1] === 'themes') {
143 location = 'themes';
144 theme = folderPath.split('/')[2];
145 }
146
147 return [workspace.workspaceFolders[0].uri.path, 'design', location, theme, 'templates', link]
148 .filter(entry => entry.trim() !== '')
149 .join('/');
150 }
```

# Plans for Capabilities

- Snippets
- Hooks
- Show all available parameters
- Go to style
- Go to script plugin
- Core ↔ Addons

# Debugging

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** extension.ts — helloworld-sample
- Toolbar:** Includes icons for Run Extension, Breakpoints, and other debugging controls.
- Left Sidebar:** Shows icons for File, Find, Replace, and Debug. The Debug icon has a blue circle with the number 2, indicating two breakpoints.
- Variables View:** Shows a Local scope with a breakpoint at line 21. The variable `message` is highlighted and set to "Hello VS Code".
- Call Stack View:** Paused on a breakpoint, showing the stack trace:
  - vscode.commands.registerCommand ...
  - e.\_executeContributedCommand ext...
  - e.\$executeContributedCommand ext...
- Code Editor:** The `extension.ts` file is open, showing the following code:

```
// The command has been defined in the package.json file
// Now provide the implementation of the command with registerCommand
// The commandId parameter must match the command field in package.json
let disposable = vscode.commands.registerCommand('extension.helloWorld',
 // The code you place here will be executed every time your command is triggered

 // Display a message box to the user
 const message = 'Hello VS Code'; message = "Hello VS Code"
 vscode.window.showInformationMessage(message); vscode = Object {version: "1.52.3", ...}

);
context.subscriptions.push(disposable);

// this method is called when your extension is deactivated
export function deactivate() {}
```
- Status Bar:** Shows the URL <code.visualstudio.com/api/get-started/your-first-extension>.

# Tips

- VSCode API docs
- Explore other repositories
- Explore extensions

# Packaging

# Package preparation

1. `yarn global add vsce`

2. `package.json:`

```
"publisher": "andreivoronin", ERROR Missing publisher name.
```

```
"license": "SEE LICENSE IN LICENSE.txt", warning package.json: No license field
```

```
"repository": { WARNING A 'repository' field is missing
```

```
 "type": "git",
```

```
 "url": "https://github.com/andreivoronin/cscartcode.git"
```

```
}
```

3. `README.md:` **ERROR** Make sure to edit the `README.md` file before you package...

~~This is the README for your extension~~

# Packaging

- vsce package

```
Executing prepublish script
'npm run vscode:prepublish'...
```

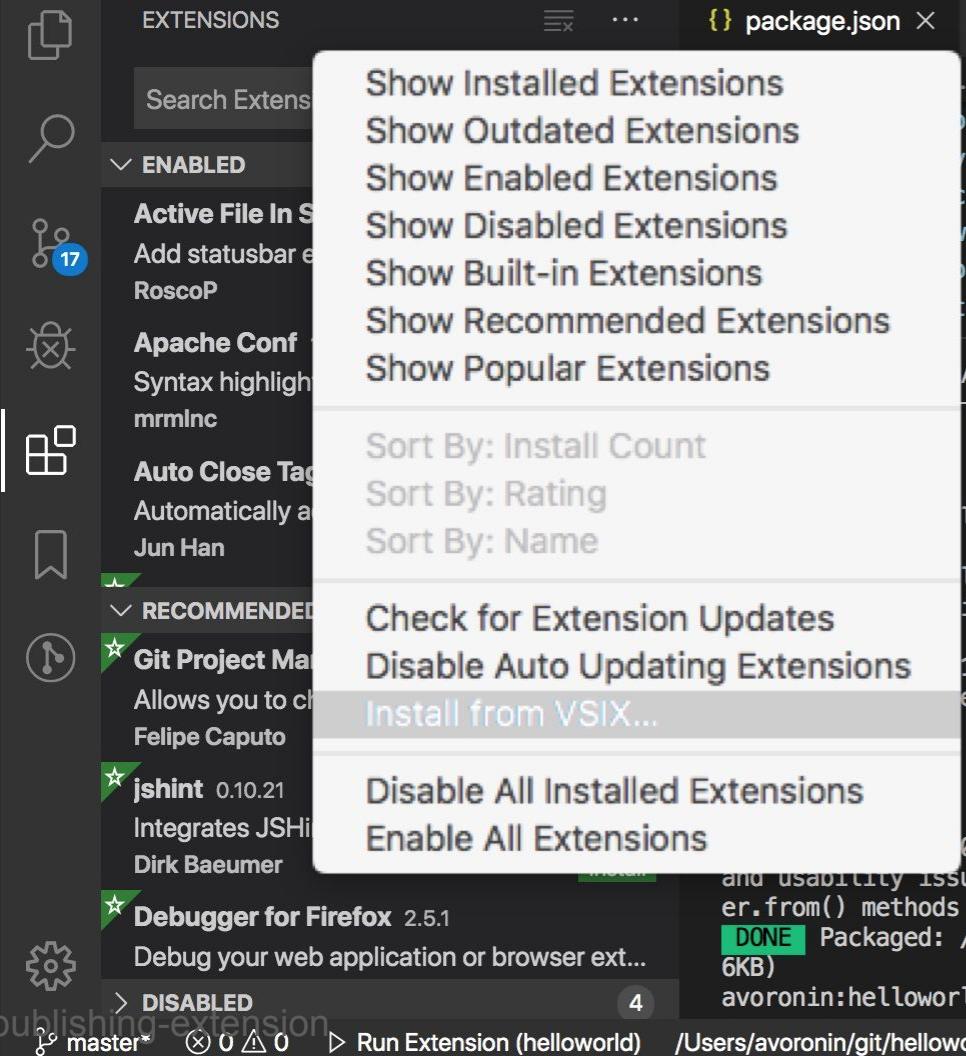
```
> helloworld@0.0.1 vscode:prepublish
/Users/avoronin/git/cscartcode
> yarn run compile
```

```
yarn run v1.3.2
$ tsc -p ./
Done in 3.59s.
```

```
DONE Packaged: /Users/avoronin/git/cscartcode/
helloworld-0.0.1.vsix (6 files, 3.6KB)
```

# Install

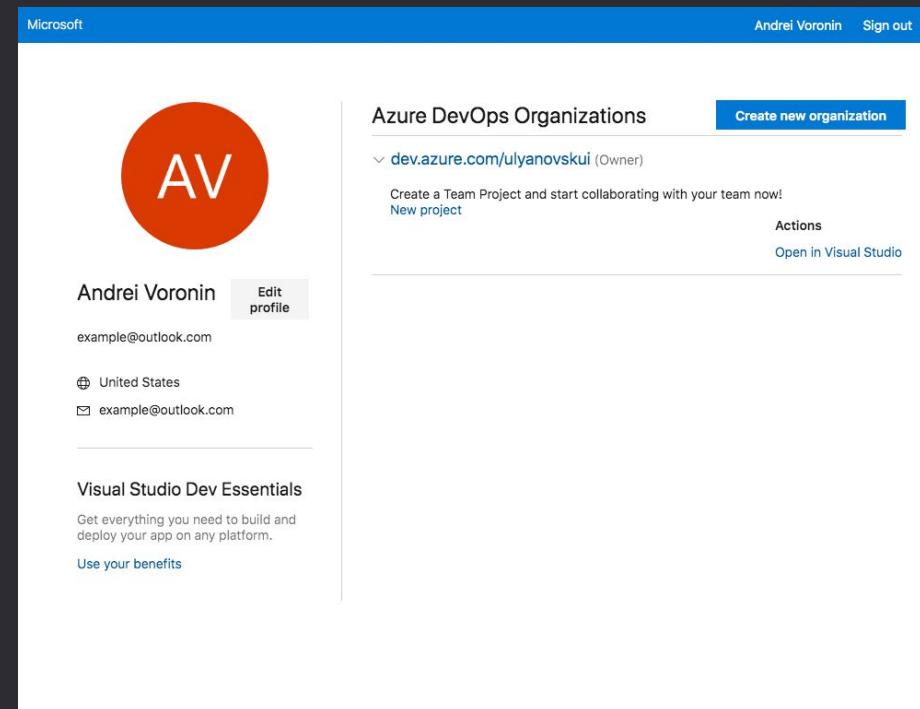
- Command Palette ⌘P  
Install from VSIX
- Extensions:  
[...] → Install from VSIX...



# Publishing

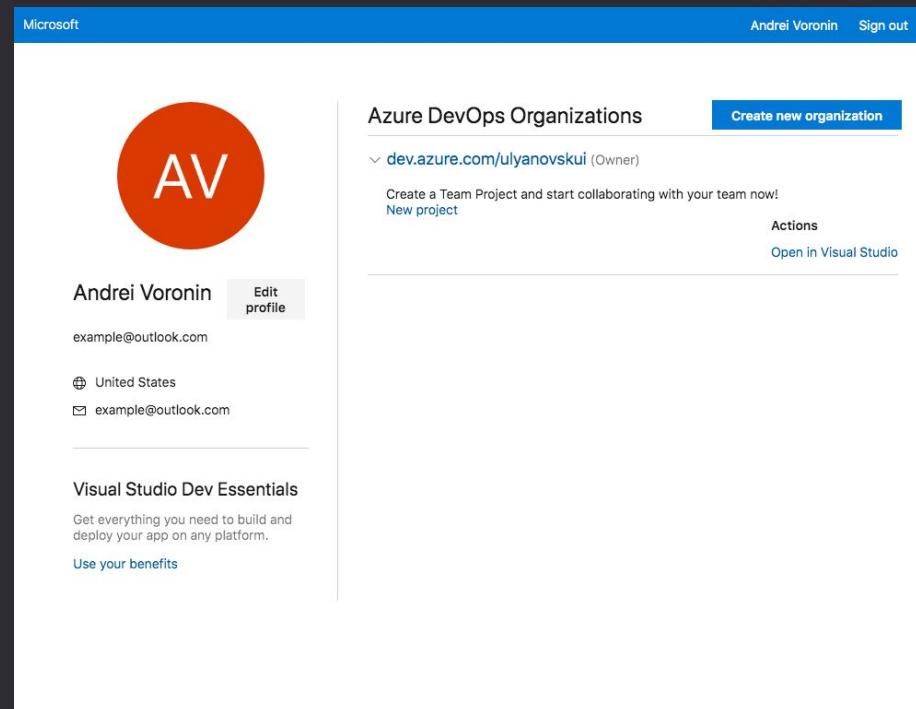
# Create an organization

1. Sign in to Azure DevOps  
[app.vsaex.visualstudio.com](http://app.vsaex.visualstudio.com)



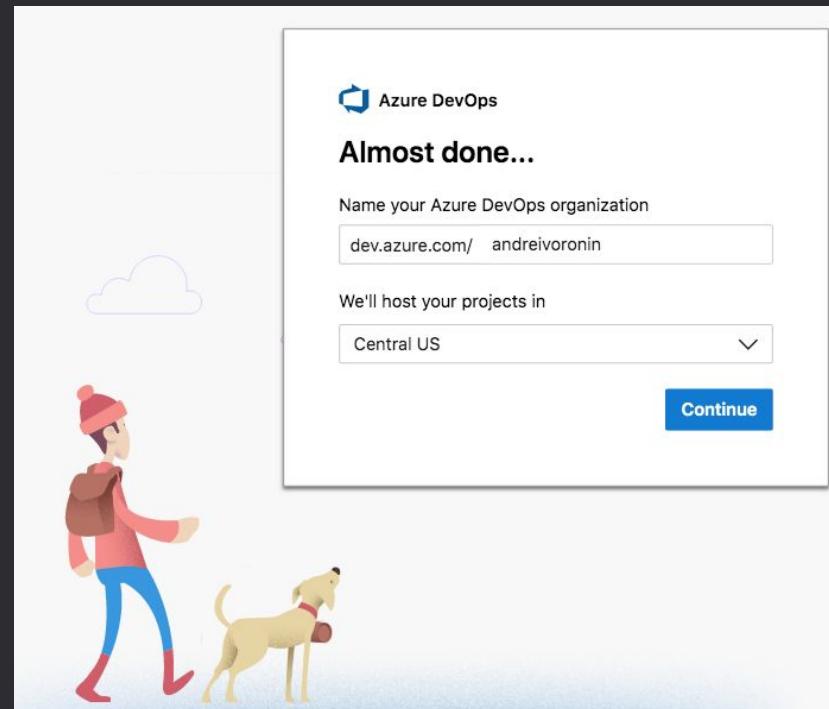
# Create an organization

1. Sign in to Azure DevOps  
[app.vsaex.visualstudio.com](http://app.vsaex.visualstudio.com)
2. Create new organization



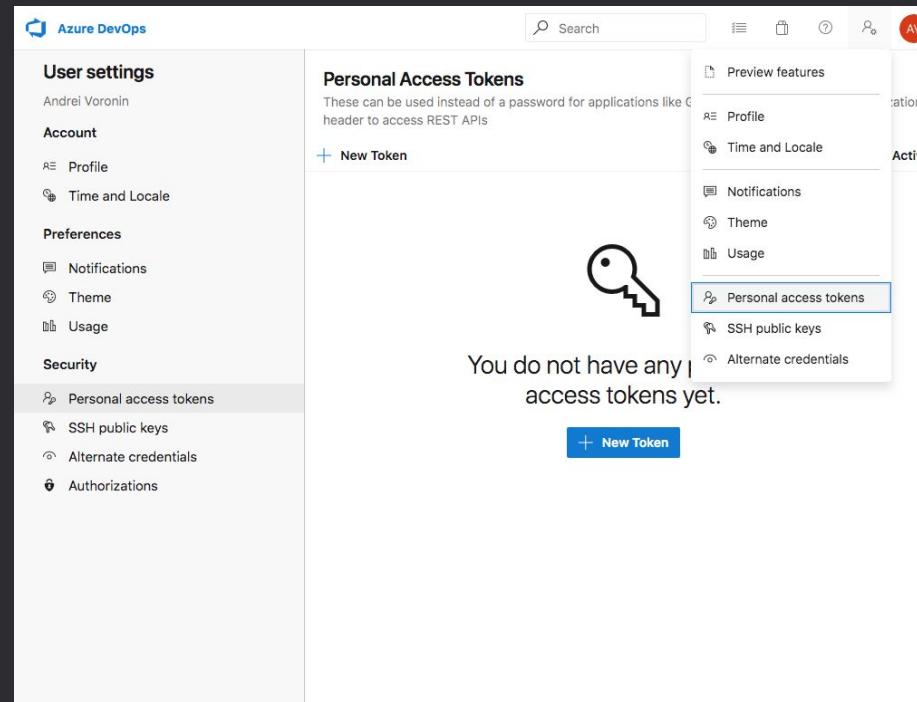
# Create an organization

1. Sign in to Azure DevOps  
[app.vsaex.visualstudio.com](http://app.vsaex.visualstudio.com)
2. Create new organization
3. [Sign in to dev.azure.com/  
myorganization](http://dev.azure.com/myorganization)



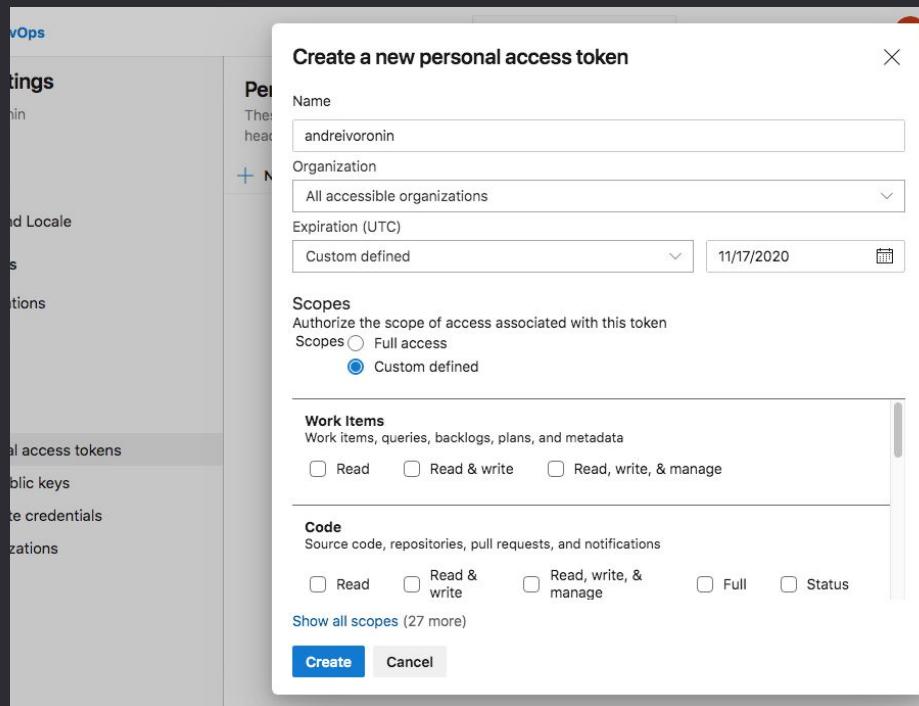
# Get a token

1. User settings →  
Personal access tokens →  
New Token



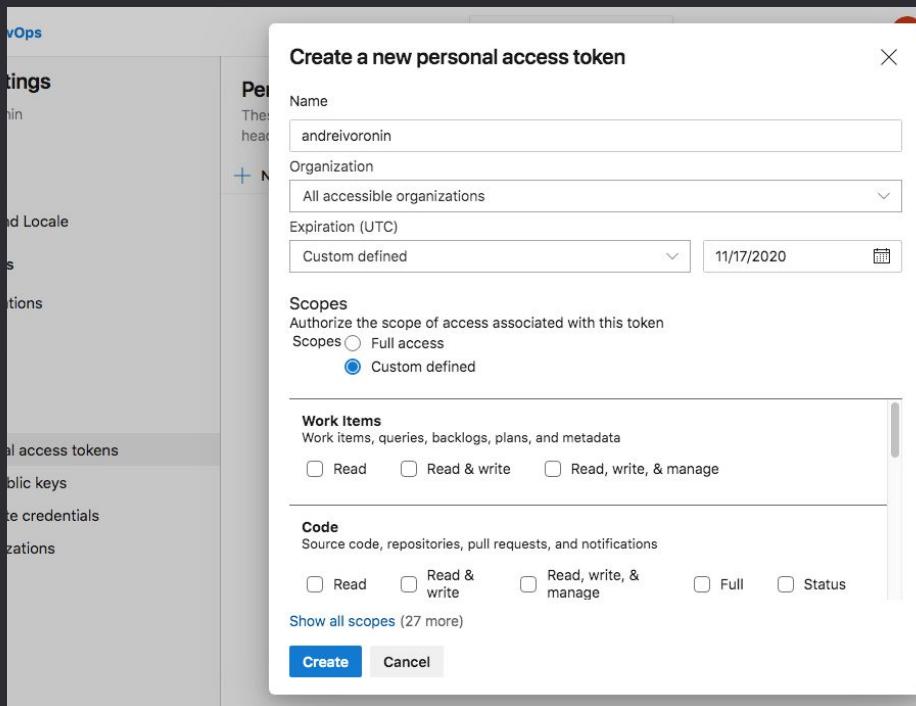
# Get a token

1. User settings →  
Personal access tokens →  
New Token
2. Name, Organization



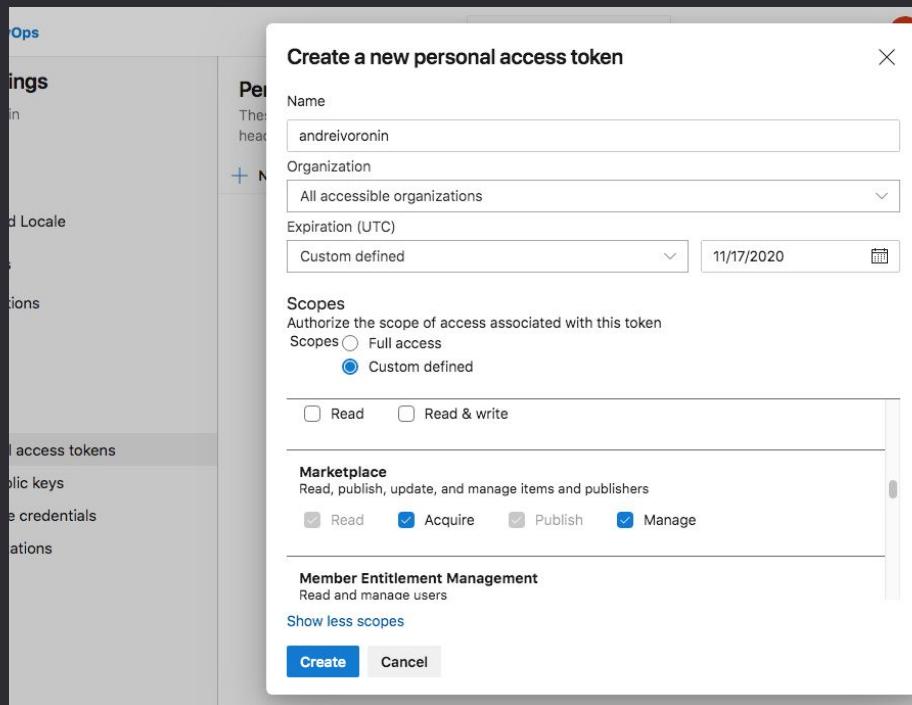
# Get a token

1. User settings → Personal access tokens → New Token
2. Name, Organization
3. Scopes: custom defined



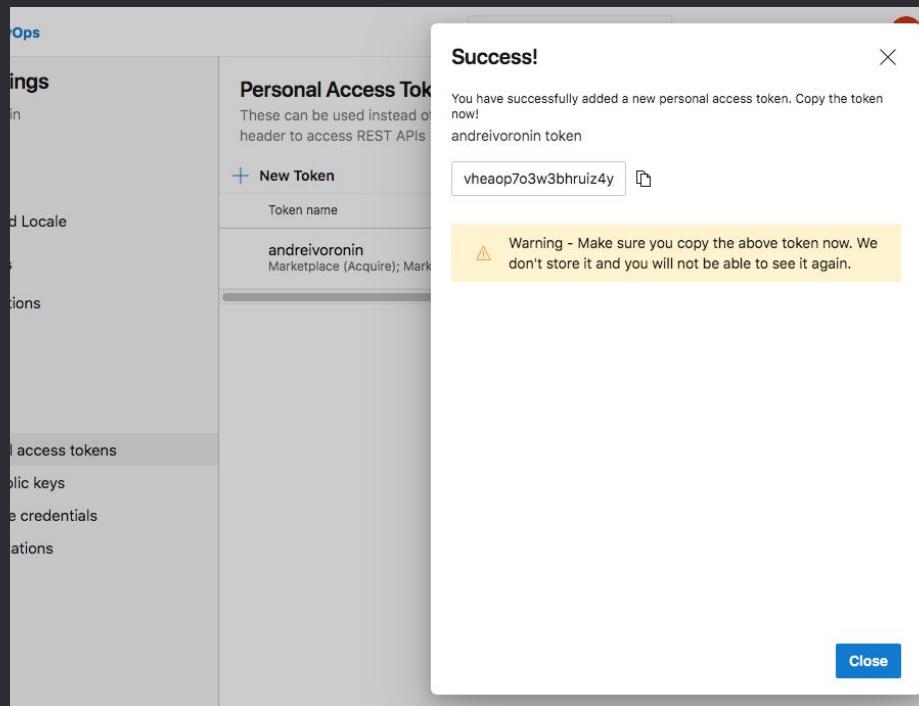
# Get a token

1. User settings →  
Personal access tokens →  
New Token
2. Name, Organization
3. Scopes: custom defined
4. Show all scopes →  
Marketplace:  
[v] Acquire, [v] Manage



# Get a token

1. User settings →  
Personal access tokens →  
New Token
2. Name, Organization
3. Scopes: custom defined
4. Show all scopes →  
Marketplace:  
[v] Acquire, [v] Manage
5. Create → Copy



# Create a publisher

1. `vsce create-publisher`  
(publisher name)
2. `vsce login` (publisher name)
3. Do you want to overwrite its  
PAT? y

```
vsce create-publisher andreivoronin
Publisher human-friendly name: (testcomp) Andrei Voronin
E-mail: ulyanovskui@example.com
Personal Access Token:

```

**DONE** Created publisher 'andreivoronin'.

```
vsce login andreivoronin
Publisher 'andreivoronin' is already known
Do you want to overwrite its PAT? [y/N] y
Personal Access Token for publisher 'andreivoronin':

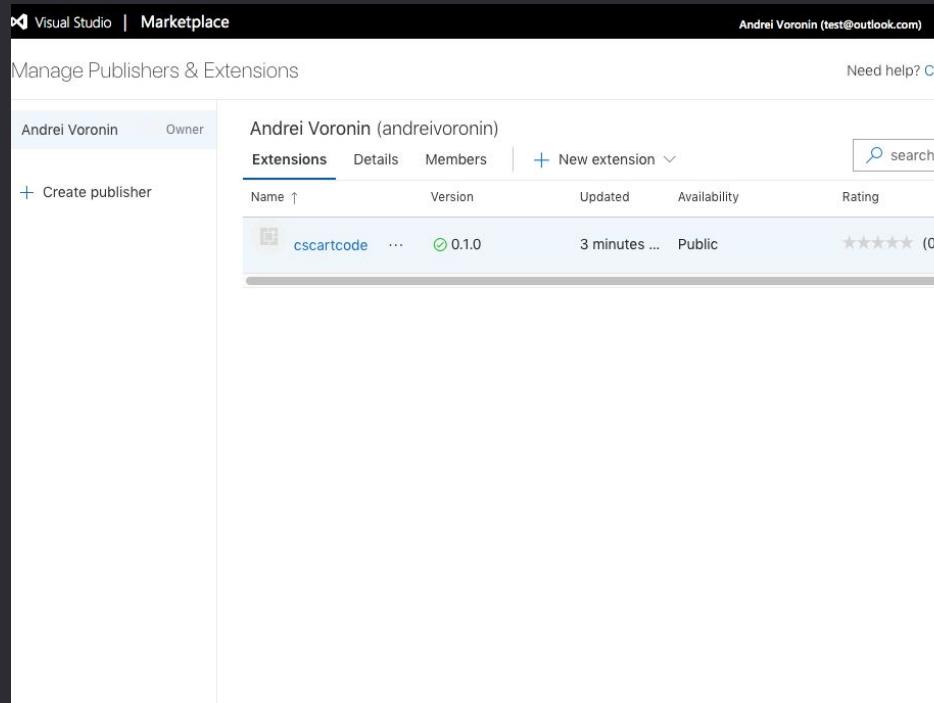
```

# Publishing

## 1. vsce publish minor

a. SemVer

b. major, minor, or patch



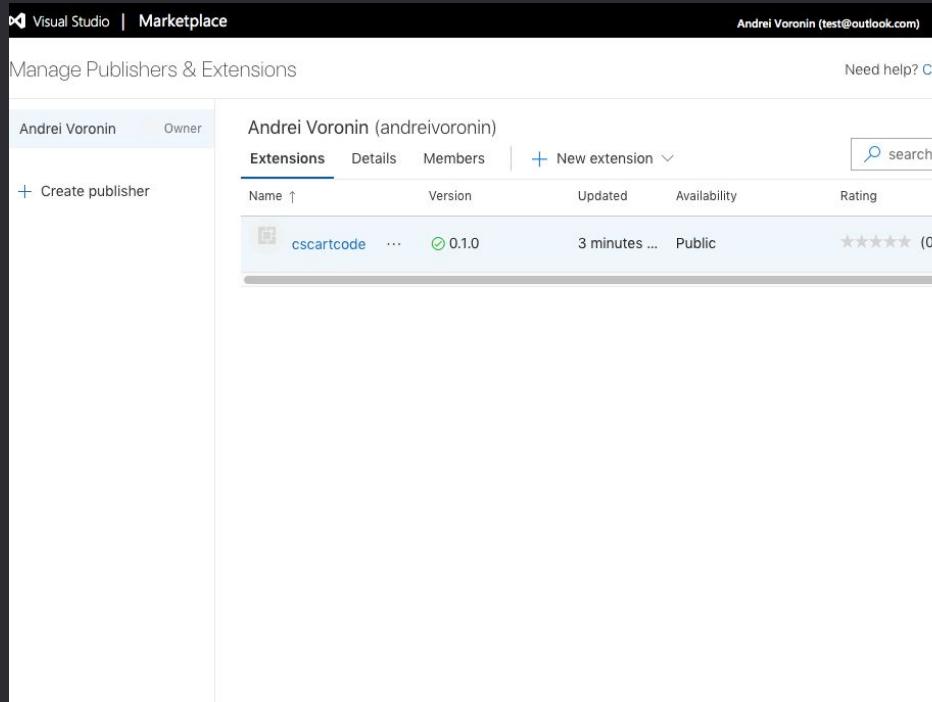
# Publishing

1. vsce publish minor

a. SemVer

b. major, minor, or patch

2. marketplace.visualstudio.com/manage



# Publishing

1. `vsce publish minor`

a. SemVer

b. major, minor, or patch

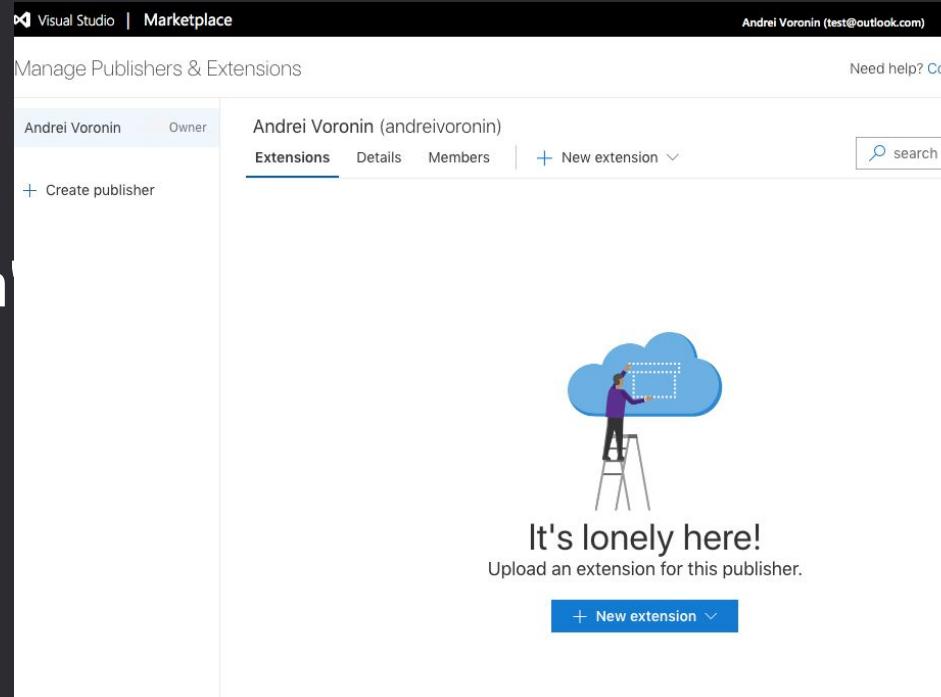
2. [marketplace.visualstudio.com/manage](https://marketplace.visualstudio.com/manage)

3. [marketplace.visualstudio.com/items?  
itemName=andreivorонин.cscartcode](https://marketplace.visualstudio.com/items?itemName=andreivorонин.cscartcode)

The screenshot shows the Visual Studio Marketplace page for the 'cscartcode' extension. At the top, the navigation bar includes 'Visual Studio | Marketplace', 'Sign in', and 'New to Visual Studio?'. The breadcrumb path 'Visual Studio Code > Other > cscartcode' is shown. The extension card for 'cscartcode' by Andrei Voronin has a 5-star rating (0 reviews) and is labeled as 'Free'. A prominent green 'Install' button is at the top left, followed by a 'Trouble Installing?' link. Below the card, there are tabs for 'Overview' (which is selected), 'Q & A', and 'Rating & Review'. The 'Overview' section contains the 'cscartcode README' which states: 'This is the README for your extension "cscartcode". After writing up a brief description, we recommend including the following sections.' It also includes a 'Features' section with instructions for describing specific features and including screenshots. On the right side, there are 'Categories' (Other), 'Resources' (Repository, Changelog, Download Extension), 'Project Details' (GitHub link), and 'More Info' (Version 0.1.0).

# Unpublishing

1. `vsce unpublish  
publishername.myextension`
2. This will FOREVER delete  
'publishername.myextension'  
Are you sure? [y/N] y

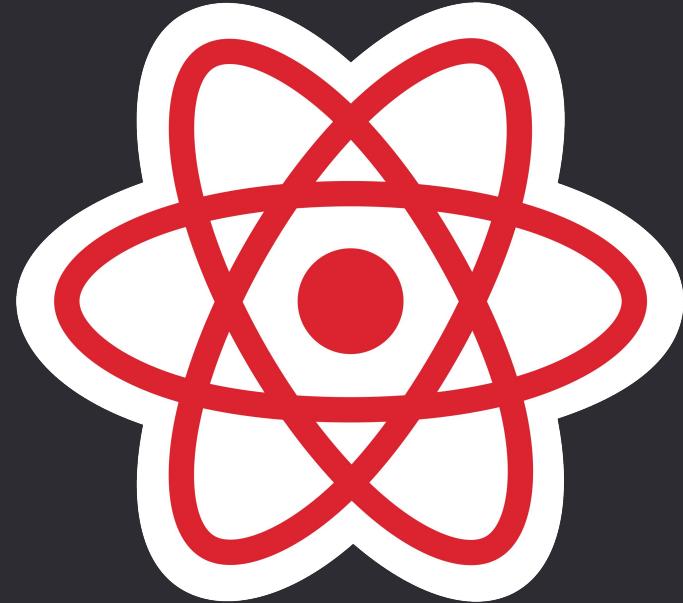


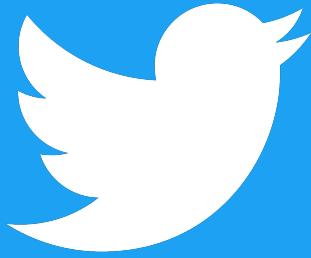
# Results

- Run & Debug
- VSCode API
- Packaging & Publishing
- Accelerate development

simtech

# Middle React Developer





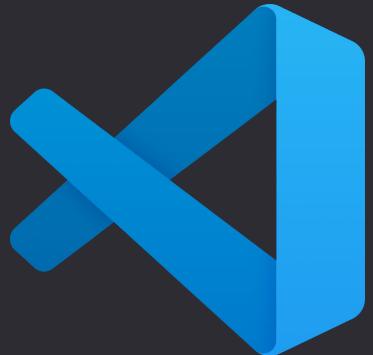
# @UlyanovskUI

Product, Frontend & Design

400 FOLLOWERS

30K /m TWEETS SHOW

8 YEARS



[bit.ly/firstvscode](https://bit.ly/firstvscode)

Andrei Voronin  UlyanovskUI

# Your First VS Code Extension

## Andrei Voronin @UlyanovskUI

- 2. About
- 3. Accelerate development
- 5. Visual Studio Code
- 8. Capabilities
- 9. Hello world
- 16. Structure
- 19. Anatomy
- 20. CMS
- 23. Language variables
- 31. Include
- 41. Plans for Capabilities
- 42. Debugging
- 43. Tips
- 44. Packaging
- 47. Install
- 48. Publishing
- 61. Unpublishing
- 62. Results
- 63. Simtech
- 64. @UlyanovskUI