



DOCUMENTS

Visual Studio Code

DOCUMENTS
Extension

Christine Heller

Inhalt

Tools mit PortalScripting Unterstützung

Einführung in den Visual Studio Code

DOCUMENTS Extension

Live-Demos

Tools mit PortalScripting Unterstützung

- Notepad++
 - Einfacher Editor
 - DOCUMENTS 4 und 5
- Eclipse
 - Umfangreiche Entwicklungsumgebung
 - DOCUMENTS 4 und 5
- **Visual Studio Code**
 - Kombination: einfach mit guten Entwicklungswerkzeugen
 - DOCUMENTS 5.0b/c (DOCUMENTS 4 abgekündigt zum 31.12.2018)
 - Feature teilweise abhängig von der DOCUMENTS Version

Einführung in den Visual Studio Code

Visual Studio Code

Allgemein

- Kostenloser Open Source Editor von Microsoft
- Verfügbar für Windows, Linux, Mac
- Beliebteste Tool (Stack Overflow Umfrage 2018)

Installation und erste Schritte

- Free Download
- Monatliche Updates (automatische Hinweise, einfache Installation)
- Dokumentation und Erste Schritte
- **VS Code auf Ordner öffnen! (Windows Explorer Kontextmenü)**

Übersicht

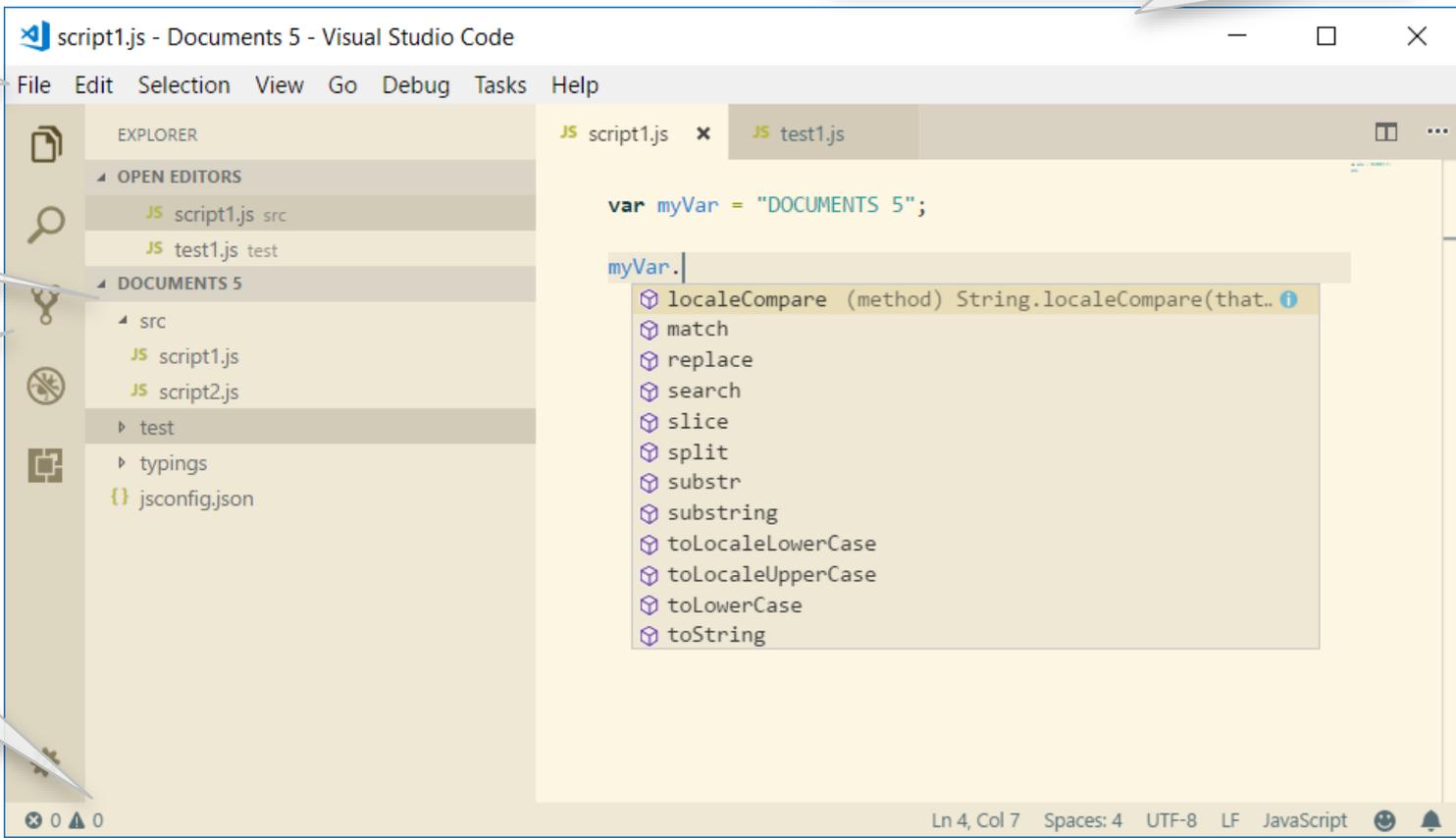
Menüzeile

Datei-Explorer

View Bar

Status Bar

Editor
Unterstützung für JavaScript integriert



Views ein- und ausschalten

Explorer
(Default-Ansicht)

Suchen in Dateien
(ausgewählt)

Quellcodeverwaltung
(Git integriert)

Debugging

Extensions verwalten
(DOCUMENTS Extension)

The screenshot shows the Visual Studio Code interface with the SEARCH view active. The title bar reads "script1.js - Documents 5 - Visual Studio Code". The menu bar includes "File", "Edit", "Selection", "View", "Go", "Debug", "Tasks", and "Help". The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The SEARCH view displays the search term "DOCUMENTS" and a single result in "script1.js" where "DOCUMENTS" is highlighted. The code editor on the right shows the code: `var myVar = "DOCUMENTS 5";` with "DOCUMENTS" highlighted in orange.

Befehle

Tipp

Erstes Zeichen bestimmt den **Modus**

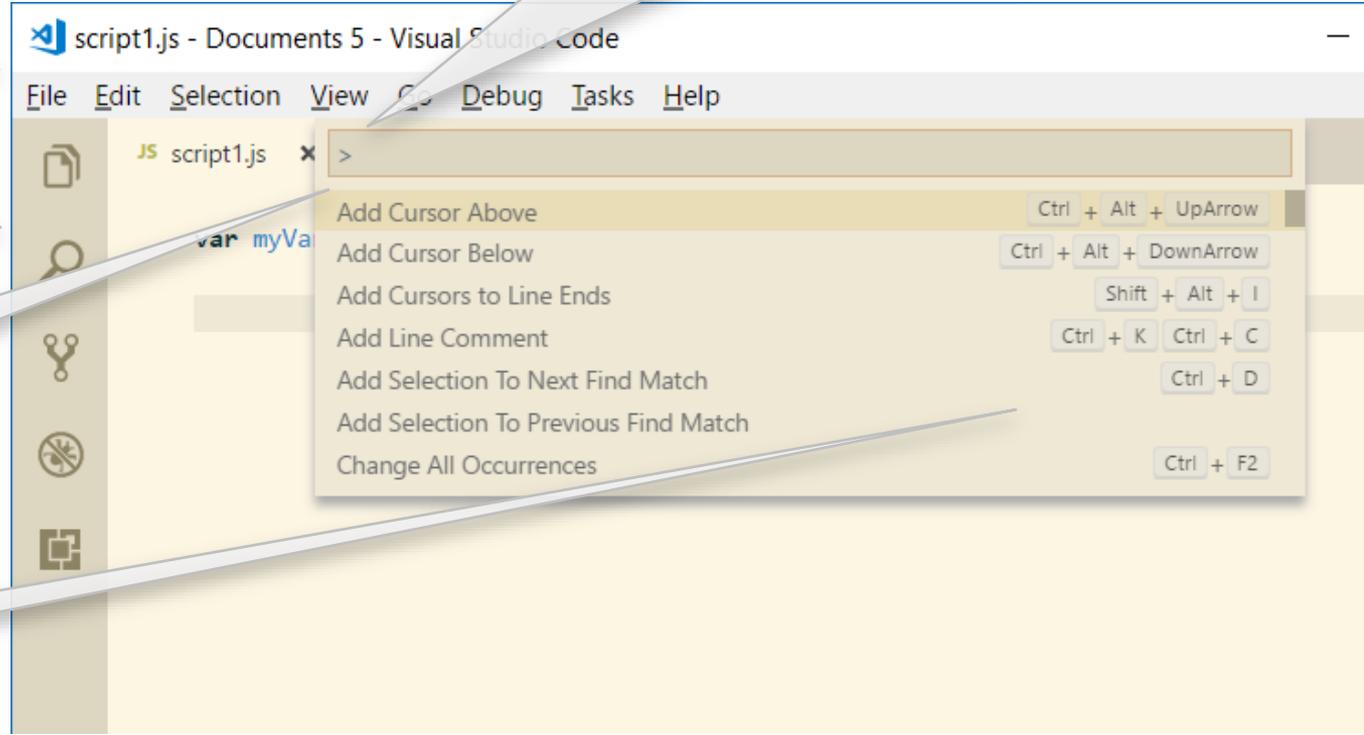
- > **Befehle** anzeigen/ausführen
- @ **Symbole** anzeigen/hinspringen
- **Dateien** anzeigen/öffnen
- ? **Modi** anzeigen/wechseln

Menüzeile enthält die üblichen Editor-Befehle

Explorer und Editor enthalten Befehle in Kontextmenüs

Kommando-Palette (F1) enthält alle Befehle

Shortcuts für viele Befehle



Shortcuts

- Übersicht über alle Shortcuts
- Kommando-Palette
 - Befehle anzeigen/ausführen **F1**
 - Dateien anzeigen/öffnen **Ctrl + P**
 - Symbole der geöffneten Datei (sortiert) **Ctrl + Shift + O (: in Palette)**
 - Symbole aus allen Dateien **Ctrl + T (first Letter in Palette)**
 - Go to last/next position **Alt + Left/Right Arrow**
- Editor
 - Suggestions (Code Completion) **Ctrl + Space**
 - Kommentar einfügen/entfernen **Ctrl + #**
- Shortcuts
 - Shortcuts ändern **Ctrl + K S**

Konfiguration

- Keine Menüs oder Dialogfenster
- Konfigurationsdateien (JSON) direkt bearbeiten
 - VS Code bietet Informationen und Hilfen bei der Eingabe
- settings.json
 - Allgemeine Einstellungen
- keybindings.json
 - Shortcuts konfigurieren
- launch.json
 - Einstellungen für Debugging
- jsconfig.json
 - Einstellungen für JavaScript IntelliSense (Code Completion)

JavaScript Unterstützung

- Syntax Highlighting, Bracket Matching
- Code Navigation
 - Go to Definition
 - Go to last Position
- Refactoring
 - Code Bereich als Funktion extrahieren
 - Symbol an allen Stellen umbenennen
- **IntelliSense**
 - **Code Completion**, Member Listen, Parameter Info, Quick Info
 - PortalScripting-API: Projekt konfigurieren (Wizard!)

VS Code DOCUMENTS Extension

Name

- JavaScript Remote Debugger for JANUS Apps

Technischer Name

- `vscode-janus-debug`

DOCUMENTS Extension Dokumentation

Dokumentation

- Auf GitHub
 - [Wiki](#), [README](#) und [CHANGELOG](#)
- View Bar → Extensions → JavaScript Remote Debugger for JANUS Apps
 - README und CHANGELOG

Befehle und Settings

- Liste in README
- Kommando-Palette → Filter/Präfix: DOCUMENTS
- `settings.json` → Filter/Präfix: `vscode-janus-debug`

DOCUMENTS Extension installieren

1. Bereich für
Erweiterungen
auswählen

2. **otris** eintippen

3. Install

The screenshot shows the Visual Studio Code interface with the Extensions Marketplace open. The search bar contains the text "otris". The search results list the extension "JavaScript Remote Debugger for J... 0.0.32" by "otris software". An "Install" button is visible next to the extension name. The right-hand pane displays the extension's details, including an orange "JS" logo, the name "JavaScript Remote Debugger for JANUS Apps", the publisher "otris software", and a description "Debug your JANUS-based applications in V...". A yellow "Install" button is also present in the details pane. Below the details, the extension's identifier "vscode-janus-debug" is visible.

Extension: JavaScript Remote Debugger for JANUS Apps - Documents 5 - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXTENSIONS: MARKETPLACE

otris

JavaScript Remote Debugger for J... 0.0.32
Debug your JANUS-based applications in V..
otris software **Install**

JavaScript
otris software
Debug your JANUS-
Install

[Details](#) [Contributions](#) [Changelog](#) [De](#)

vscode-janus-debug

build ongoing

DOCUMENTS Extension installieren

1. Bereich für
Erweiterungen
auswählen

2. otris eintippen

3. Install

4. Reload

The screenshot shows the Visual Studio Code interface with the 'EXTENSIONS: MARKETPLACE' view. The search bar contains 'otris', and the search results list the 'JavaScript Remote Debugger for JANUS Apps' extension by 'otris software'. The extension details panel on the right shows the extension's logo (an orange square with 'JS'), the name 'JavaScript Remote Debugger for JANUS Apps', the version '0.0.32', and the publisher 'otris software'. Below the details, there are buttons for 'Reload' and 'Uninstall'. The bottom of the details panel shows the extension's identifier 'vscode-janus-debug' and a 'Build' button.

Extension: JavaScript Remote Debugger for JANUS Apps - Documents 5 - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXTENSIONS: MARKETPLACE

otris

JavaScript Remote Debugger for J... 0.0.32
Debug your JANUS-based applications in V..
otris software **Reload** ⚙️

JavaScript
otris software
Debug your JANUS-
Reload **Uninstall**

[Details](#) [Contributions](#) [Changelog](#) [De](#)

vscode-janus-debug

Build **Uninstall**

DOCUMENTS Extension installieren

1. Bereich für
Erweiterungen
auswählen

2. otris eintippen

3. Install

4. Reload

Updates werden
automatisch
installiert!

The screenshot shows the Visual Studio Code interface with the 'EXTENSIONS: MARKETPLACE' view. The search bar contains 'otris'. The search results show the extension 'JavaScript Remote Debugger for J...' by 'otris software' with version '0.0.32'. The extension's details are displayed on the right, including the 'JS' logo and buttons for 'Disable' and 'Uninstall'. The extension name 'vscode-janus-debug' is visible at the bottom.

Extension: JavaScript Remote Debugger for JANUS Apps - Documents 5 - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXTENSIONS: MARKETPLACE

otris

JavaScript Remote Debugger for J... 0.0.32
Debug your JANUS-based applications in V..
otris software

JavaScript
otris software |
Debug your JANUS-
Disable Uninstall

[Details](#) [Contributions](#) [Changelog](#) [De](#)

vscode-janus-debug

build passing

Erste Schritte

Scripte aus dem Manager mit VS Code „extern bearbeiten“

- Umgebungsvariable setzen
 - `SCRIPTEDITOR = C:\Program Files (x86)\Microsoft VS Code\Code.exe -w -n`
- Nur editieren, keine Features der DOCUMENTS Extension!

Arbeiten im „Projekt“

- Projekt mit Wizard erstellen
- Scripte bearbeiten
 - Extension: Dokumentation, Befehle und Settings
 - PortalScripting API: IntelliSense, Dokumentation

Wizard

1. Windows Explorer Kontextmenü auf leerem Ordner → mit VS Code öffnen
2. Befehl **Wizard: Download / Create Project**
3. Unterordner für Kategorien erstellen?
4. Credentials eingeben
 - launch.json
5. Browser auswählen
 - → View Documentation in Browser
6. Projekt wird erstellt
 - Alle PortalScripte des DOCUMENTS Servers im Ordner src

Credentials eingeben

Der erste Befehl, der eine Verbindung zum DOCUMENTS Server aufbaut, fragt alle benötigten Informationen ab

Hostname und Port des Servers

Mandant

Benutzername (Redakteur):

- admin oder
- **Benutzername und Mandant getrennt durch einen Punkt**

Passwort

localhost

Please enter the hostname (Press 'Enter' to confirm or 'Escape' to cancel)

11000

Please enter the port (Press 'Enter' to confirm or 'Escape' to cancel)

relations

Please enter the principal (Press 'Enter' to confirm or 'Escape' to cancel)

admin

Please enter the username (username.principal) (Press 'Enter' to confirm or 'Escape' t...

schreiber.relations

Please enter the username (username.principal) (Press 'Enter' to confirm or 'Escape' t...

|

Please enter the password (Press 'Enter' to confirm or 'Escape' to cancel)

Live Demo

Projekt mit Wizard

Befehle und Settings

Wizard – Projektstruktur

Anmelde-Informationen
Konfiguration / [launch.json](#)

Allgemeine Einstellungen
Konfiguration / [settings.json](#)

Alle Scripte
Kategorien: [Wiki](#), [Folie](#)

IntelliSense Metadaten
[IntelliSense](#), [*.d.ts](#)

Extension Informationen
[.vscode-janus-debug](#)

IntelliSense Projektinformationen
[IntelliSense](#) / [jsconfig.json](#)

The screenshot shows the Visual Studio Code interface with the Explorer view on the left and the Editor view on the right. The Explorer view displays the following structure:

- .vscode
 - launch.json
 - settings.json
- src
- typings
 - fileTypes.d.ts
 - portalScripting.d.ts
 - scriptExtensions.d.ts
- .vscode-janus-debug
- jsconfig.json (selected)

The Editor view shows the content of `jsconfig.json`:

```
1 {
2   "include": [
3     "src/**/*",
4     "typings/**/*"
5   ]
6 }
7
```

settings.json

Default Settings mit
Doku und Suche

User und Workspace
Settings umschaltbar

Dokumentation
anzeigen

File →
Preferences →
Settings:

Default-, User-
und Workspace-
Settings

The screenshot shows the Visual Studio Code settings editor for the extension 'vscode-janus-debug'. The editor is split into two panes. The left pane shows the 'Workspace Settings' section with default settings for the extension. The right pane shows the 'User Settings (1)' and 'Workspace Settings (2)' tabs, which are circled in orange. The 'Workspace Settings (2)' tab is active, showing a list of settings for the extension, including 'vscode-janus-debug.categories', 'vscode-janus-debug.browser', 'vscode-janus-debug.encryptionOnUpload', 'vscode-janus-debug.forceUpload', 'vscode-janus-debug.log', 'vscode-janus-debug.scriptLog', 'vscode-janus-debug.scriptParameters', 'vscode-janus-debug.serverConsole', 'vscode-janus-debug.uploadManually', 'vscode-janus-debug.uploadOnSave', and 'vscode-janus-debug.uploadOnSaveGlobal'. The 'vscode-janus-debug.categories' setting is highlighted in yellow, and the 'vscode-janus-debug.browser' setting is highlighted in orange. The 'vscode-janus-debug.encryptionOnUpload' setting is highlighted in blue. The 'vscode-janus-debug.uploadManually' setting is highlighted in green. The 'vscode-janus-debug.uploadOnSave' setting is highlighted in red. The 'vscode-janus-debug.uploadOnSaveGlobal' setting is highlighted in purple.

```
settings.json - Documents 5 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

Workspace Settings
vscode-janus-debug 11 Settings found

DEFAULT SETTINGS
Place your settings in the right hand side editor to override.

// Specifies the browser for 'View Documentation In
Browser'. (Based on
https://github.com/pwnall/node-open)
"vscode-janus-debug.browser": "",

// Project folders with postfix '.cat' will be
created and deleted on up- or download!
"vscode-janus-debug.categories": false,

// Default: script will be encrypted, if it's
encrypted on server or if it contains the crypt
keyword
"vscode-janus-debug.encryptionOnUpload": "default",

// Set to true, if the extension should upload
scripts without checking the script on server.

USER SETTINGS (1) WORKSPACE SETTINGS (2)
Place your settings here to overwrite the User Settings.
{
  "vscode-janus-debug.categories": true,
  "vscode-janus-debug.browser": "chrome",
  "vscode-janus-debug.
  vscode-janus-debug.encryptionOnUpload
  vscode-janus-debug.forceUpload
  vscode-janus-debug.log
  vscode-janus-debug.scriptLog
  vscode-janus-debug.scriptParameters
  vscode-janus-debug.serverConsole
  vscode-janus-debug.uploadManually
  vscode-janus-debug.uploadOnSave
  vscode-janus-debug.uploadOnSaveGlobal
```

launch.json

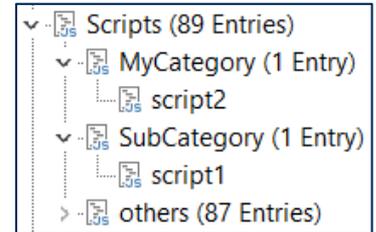
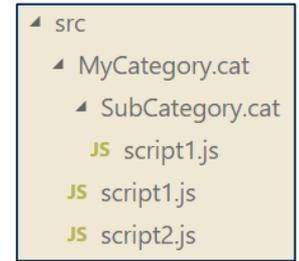
- Credentials werden in der launch.json gespeichert
- Passwort (plain!) speichern ist optional
 - Ansonsten Abfrage einmal nach Starten der Extension
- launch.json kann manuell geändert oder erstellt werden
 - Änderungen werden sofort übernommen
 - Benutzername beachten
- Credentials über Eingabebox eingeben
 - → launch.json löschen

```
{ } launch.json x
```

```
{  
  // Use IntelliSense to learn about possible  
  // Hover to view descriptions of existing  
  // For more information, visit  
  // https://github.com/otris/vscode-janus-de  
  "version": "0.0.32",  
  "configurations": [  
    {  
      "name": "Launch Script on Server",  
      "request": "launch",  
      "type": "janus",  
      "script": "",  
      "username": "admin",  
      "password": "",  
      "principal": "relations",  
      "host": "localhost",  
      "applicationPort": 11000,  
      "debuggerPort": 8089,  
    }  
  ]  
}
```

Kategorien als Unterordner

- `settings.json: vscode-janus-debug.categories = true`
- Wird im Wizard abgefragt!
- Download
 - Skripte aus Kategorie **MyCategory** → Unterordner **MyCategory.cat**
- Upload
 - Skripte aus Unterordner **MyCategory.cat** → Kategorie **MyCategory**
 - Warnung (optional) wenn Kategorie geändert wird
- **Kategorien ≠ Unterordner**
 - Kategorienamen können Sonderzeichen enthalten – Ordnernamen nicht
 - Kategorien haben nur eine Ebene – Ordner haben mehrere Ebenen
 - Skripte mit gleichem Namen in Unterordnern möglich – in Kategorien nicht



Live Demo

PortalScripting API:

- IntelliSense
- Dokumentation

Type Declaration Files – Typings

Allgemein

- Enthalten nur Deklarationen und Typ Informationen
- Ermöglichen IntelliSense für „fremde“ Module
- Liegen meistens im Ordner Typings

Befehl `Install IntelliSense`

- Erstellt **PortalScripting Typings** passend zum Server
- Wird vom Wizard aufgerufen!

PortalScripting Typings

`portalScripting.d.ts`

- Allgemeine PortalScripting API
- Abhängig von DOCUMENTS Version

`fileTypes.d.ts`

- Mappentypen auf dem Server
- [JSDoc Kommentare!](#)

`scriptExtensions.d.ts`

- Scriptlist, ScriptTree, Guided Tour

jsconfig.json

- Default (keine `jsconfig.json`):
 - Alle Dateien unabhängig von einander
 - Kein IntelliSense für Dateien des Arbeitsbereichs
- Mit `jsconfig.json` JavaScript Projekt definieren
 - IntelliSense aus allen Dateien des JavaScript Projekts
 - Projekt definieren über `include` Property
 - In **Install IntelliSense / Wizard** erstellt falls nicht vorhanden

```
▶ src
▶ test
▶ typings
{} jsconfig.json
```

IntelliSense
aus allen Dateien in den hier
aufgelisteten Ordnern

```
{ } jsconfig.json x
{
  "include": [
    "src/**/*",
    "typings/**/*"
  ]
}
```

JSDoc

- Typ einer Variable/Funktion in JSDoc Kommentar angeben
 - → IntelliSense an der Variable/Parameter für angegebenen Typ
- **Tipp:** über einer Funktion `/** Enter` eingeben → alle `@param` automatisch

Typ durch
Definition klar

```
var myVar = "DOCUMENTS 5";  
myVar.  
  localeCompare  
  match  
  replace  
  search  
  slice  
  split  
  substr  
  substring
```

Return-Typ von getString()
nicht dokumentiert

```
/** @type {string} */  
var myVar = anyModule.getString();  
myVar.  
  localeCompare  
  match  
  replace  
  search  
  slice  
  split  
  substr  
  substring
```

JSDoc Kommentar für
Parameter

```
/**  
 * @param {string} param  
 * @returns {string}  
 */  
function myFunction(param) {  
  param.  
    localeCompare  
    match  
    replace  
    search  
    slice  
}
```


Zusammenfassung IntelliSense

Type Declaration Files

- Enthalten Deklarationen und Typ-Informationen
- IntelliSense für externe Module

jsconfig.json

- JavaScript Projekt definieren
- IntelliSense für Dateien innerhalb des JavaScript Projekts

JSDoc

- Dokumentation und Typ-Informationen für JavaScript
- IntelliSense verbessern durch Angabe von Datentypen

PortalScripting Dokumentation

Dokumentation über Kontextmenü im Editor öffnen

- Befehl: View Documentation in Browser
- settings.json: `vscode-janus-debug.browser` setzen!
 - → Sprung direkt zu Member/Funktion
- Wird im Wizard abgefragt

```
context.createFile
```

- Upload Script
- Run Script
- Upload and Run Script
- Compare Script
- View Documentation in Browser

Index x

file:///C:/Users/Heller/.vscode/extensions/otris-software.vscode-janus-debug-0.0

Portalscript API

Search Documentations

createFile(fileType) {DocFile}

Create a new file of the specified filetype.

This function creates a new file of the given filetype. The filetype is mandatory that user possesses sufficient access rights. If the method fails, the method will fail.

If an error occurs during creation of the file the returned DocFile describes the error with getLastErrorMessage(). Note: For an EAS or EBIS store, if "@server" has been granted. In this case the returned DocFile must

- Changelog
- Script exits
- Field access methods
- Filter expressions

Fazit

VS Code ist ein schöner Editor mit...

- guten Entwicklungswerkzeugen
- DOCUMENTS Anbindung
 - Upload/Download (mit Kategorien), Run, Compare, ...
- IntelliSense für PortalScripting, Mappentypen, ...
- Integrierter Dokumentation für PortalScripting
- Integrierter Server Konsole
- weiteren Features zu Verschlüsselung, Script-Parameter, Auto-Upload

und direkt aus dem otrisLAB  ab auf die DOPaK...



VS Code
> JS Debugger



otrisLAB

JS Debugger: Zielsetzung

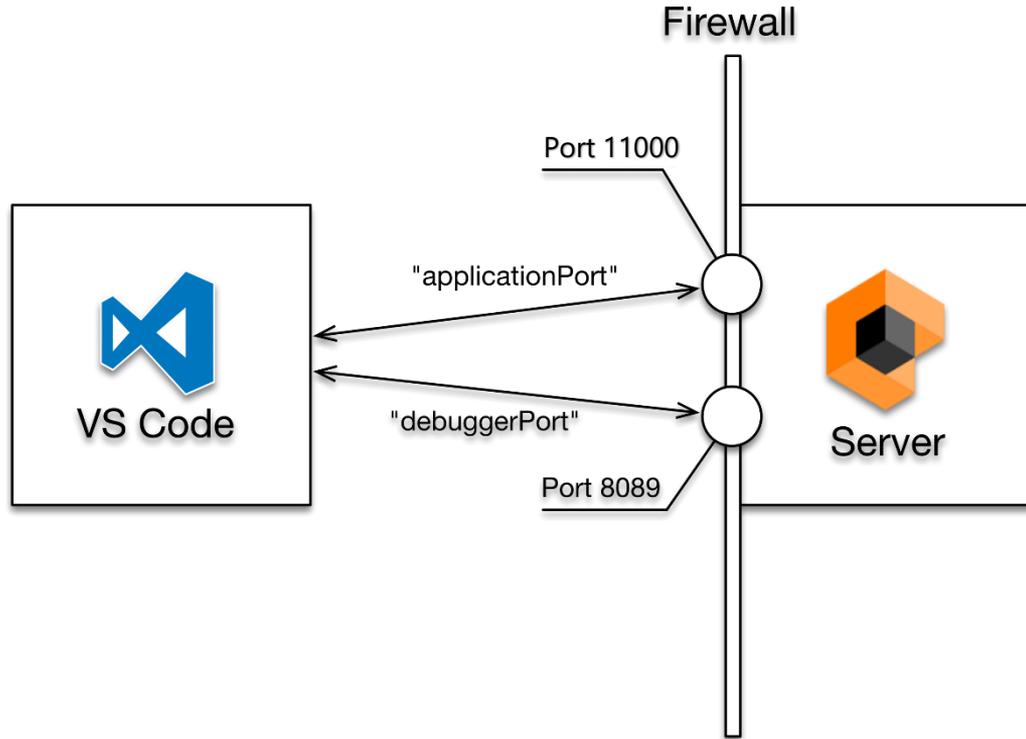
PortalScripte direkt aus dem VS Code heraus debuggen

Attach an laufende PortalScripte in DOCUMENTS 5

Wie ein „normaler“ moderner Debugger

- Während der Ausführung Breakpoints setzen
- Continue (F5)
- Next (F10)
- Step-in (F11)
- Variablen auslesen
- Support für das `debugger;` - Statement

JS Debugger: Architektur



JS Debugger: aktivieren

Unterstützt ab DOCUMENTS 5.0d

In die documents.ini eintragen

- JSDebugger yes
- [JSDebuggerPort 8089]
 - oder beliebigen anderen TCP Port, optional

Danach evtl. Firewall-Regeln und Security Groups anpassen

DOCUMENTS Server neu starten

Nicht in Live-Umgebungen verwenden!

Live Demo JS Debugger

JS Debugger: Aktueller Stand

PortalScripte aus VS Code starten

- Breakpoint setzen
- Upload und Debug Script

Nicht durch VS Code gestartete PortalScripte debuggen (attach)

- Statement **debugger** ; als „Breakpoint“ in das Script einfügen und hochladen
- Scripting-Engine stoppt implizit bei Erreichen des Statement **debugger** ;
- VS Code: Debuggen > Attach to Server
 - ggfs Context auswählen, falls mehrere Scripte laufen

JS Debugger: Kommunikationsprotokoll

```
contextId/{ JSON payload }\n
```

Request



```
1/{"type":"command","name":"step","id":"0815"}
```

```
1/{"type":"command","name":"set_breakpoint","breakpoint":{"url":"test.js","line":22},"id":"4711"}
```

Response



```
1/{"type":"info","subtype":"paused","url":"test.js","line":22,"source":"util.out(\"Hello, world\");","id":"4711"}
```

Vielen Dank!

Christine Heller

Thomas Richter

www.otris.de

otris software AG
Königswall 21
44137 Dortmund