

Building a Real World Extension



Jeff Ammons

CHIEF INSTRUCTOR CODE CAREER ACADEMY

@jeffa00 jeffa.tech

Real World Extension



What's the Problem?

Create the Project

Define Our Functions

Insert File or Image Links

Write Unit Tests

Interactively Build and Insert Figure Tag

Install Our Extension Locally



What's the Problem?

Build a Better Blog with a Static Site Generator

by Jeff Ammons

Make your blog load faster than your reader can blink with a static website generator without resorting to hand written HTML.

▶ Resume Course

Course author



Jeff Ammons

Jeff has been passionate about and active in software for over 20 years. He currently works for Sage Software as a Principal Architect and runs the Gwinnett Georgia, Microsoft User Group (GGMUG.com...

Course info

Level **Intermediate**

Rating ★★★★★ (57)

My rating ★★★★★

Duration **2h 16m**

Released **25 Nov 2015**

Bookmarked

Table of contents

Description

Exercise files

Discussion

Learning Check

Recommended

Expand all



Why Do My Blog Pages Take Forever to Load?



8m 7s



What Are the Top 10 Static Site Generators?

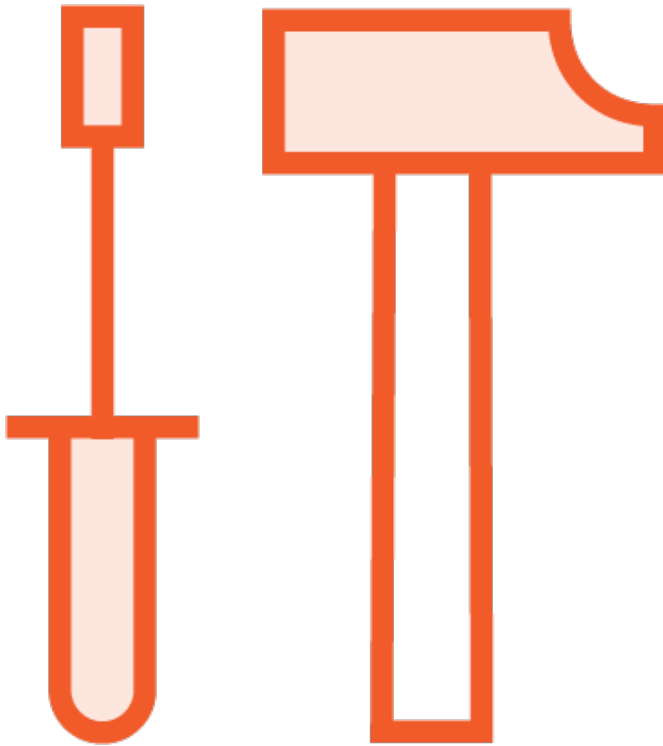


4m 35s



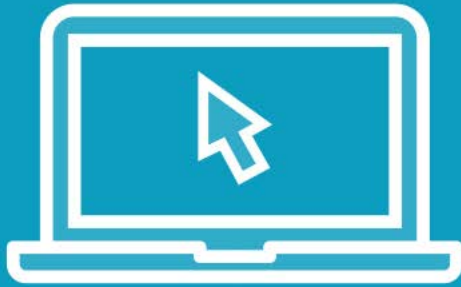


Why Not Use A Snippet?



Create the Project

Demo

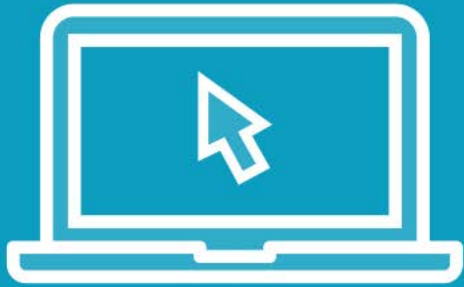


Create the Project



Define Our Functions

Demo



Define Our Functions

Functions

Insert Image or File Link

Functions

Insert Image or File Link

Insert Figure Tag

JavaScript Strict Mode

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

Tracer Bullet

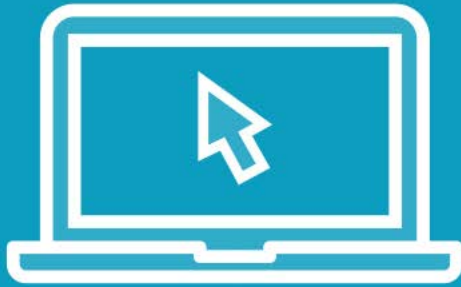
Working code of limited functionality that goes through all layers of an application.

Popularized by *The Pragmatic Programmer*



Insert File or Image Links

Demo



Insert File or Image Links



```
let fileLinkDisposable = vscode.commands.registerCommand  
('extension.insertLink', () => {  
    vscode.window.showInformationMessage('Insert File Link  
Initiated.');  
  
    insertText("Insert File Link");  
});
```


Leading Zeros

`'0' + 1 = '01' '01'`

Leading Zeros

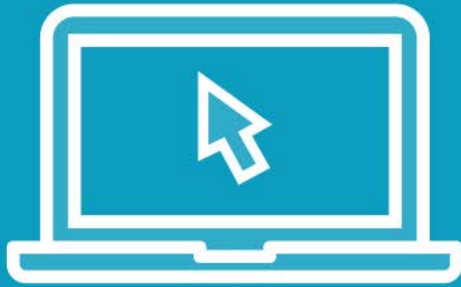
$$'0' + 1 = '01' \quad '01'$$

$$'0' + 10 = '010' \quad '10'$$



Write Unit Tests

Demo

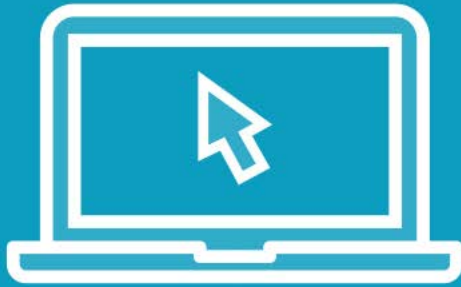


Write Unit Tests



Interactively Build and Insert Figure Tag

Demo



Interactively Build and Insert Figure Tag



Extensibility Principles and Patterns

[Edit](#)

The extension API of Visual Studio Code follows some guiding patterns and principles which are applied throughout the whole API.

Promises

The VS Code API represents asynchronous operations with [promises](#). From extensions **any** type of promise can be returned, like ES6, WinJS, A+, etc.

Being independent of a specific promise library is expressed in the API by the [Thenable](#) -type.

In most cases the use of promises is optional and when VS Code calls into an extension, it can handle the *result type* as well as a [Thenable](#) of the *result type*. When the use of a promise is optional, the API indicates this by returning [or](#) -types.

```
provideNumber(): number | Thenable<number>
```

IN THIS ARTICLE


[Promises](#)[Cancellation Tokens](#)[Disposables](#)[Events](#)[Using Node.js Modules with
Extensions](#)[Next Steps](#)[Common Questions](#)[Tweet](#)

ES6 Promise

```
vscode.window.showInputBox({prompt: "Image File Name"})  
  .then(value => {  
    }  
  .catch(error => {  
    }));
```


ES6 Promise

```
vscode.window.showInputBox({prompt: "Image File Name"})  
  .then(value => {  
    }  
  .catch(error => {  
    }));
```



Promises > Nested Callbacks

Sure Do Hope I Have Enough });s

```
1  asyncMethod01(function() {  
2    asyncMethod02(function() {  
3      asyncMethod03(function(){  
4        asyncMethod04(function(){  
5          asyncMethod05(function(){  
6            asyncMethod06(function){  
7              console.log("Hello, World!");  
8            });  
9          });  
10         });  
11       });  
12     });  
13 });
```

And Then I Said...

```
1 asyncMethod01()
2   .then(asyncMethod02)
3   .then(asyncMethod03)
4   .then(asyncMethod04)
5   .then(asyncMethod05)
6   .then(asyncMethod06)
7   .catch(function(err){
8     console.log(err);
9   });
```

***Fist bump for you,
Captain Awesome!***



showInputBox:
File Name

showInputDialog:
File Name



“Thenable”

showInputDialog:
Caption

showInputDialog:
File Name

“Thenable”

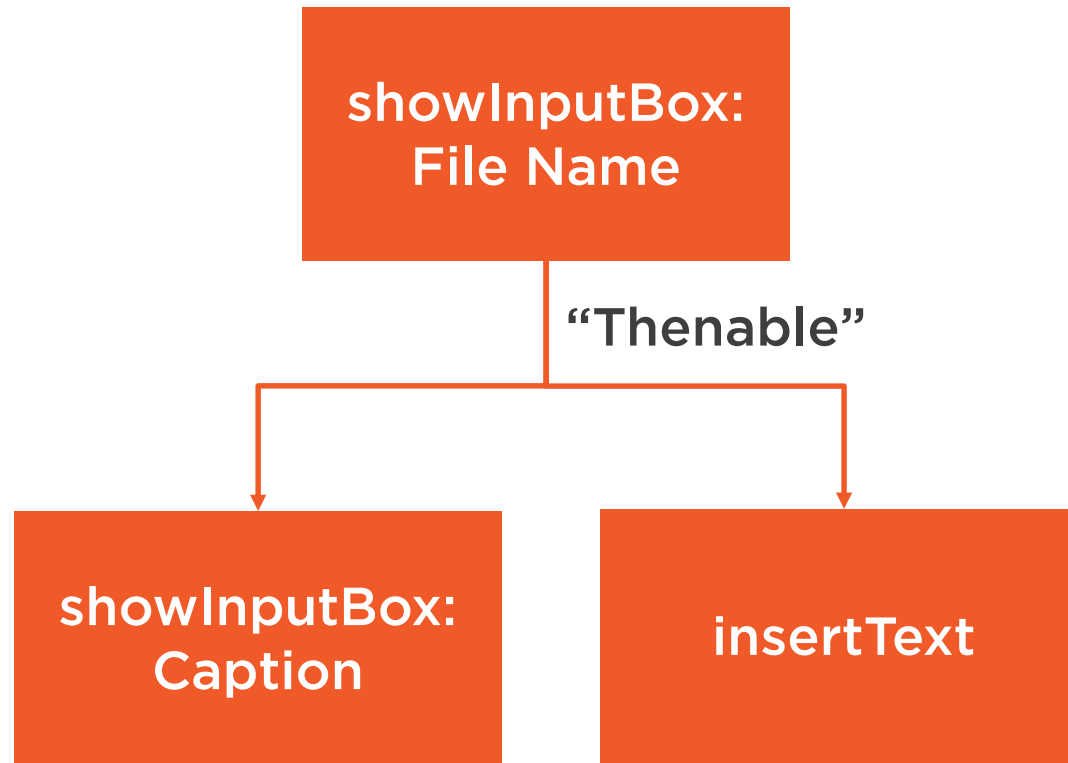
showInputDialog:
Caption

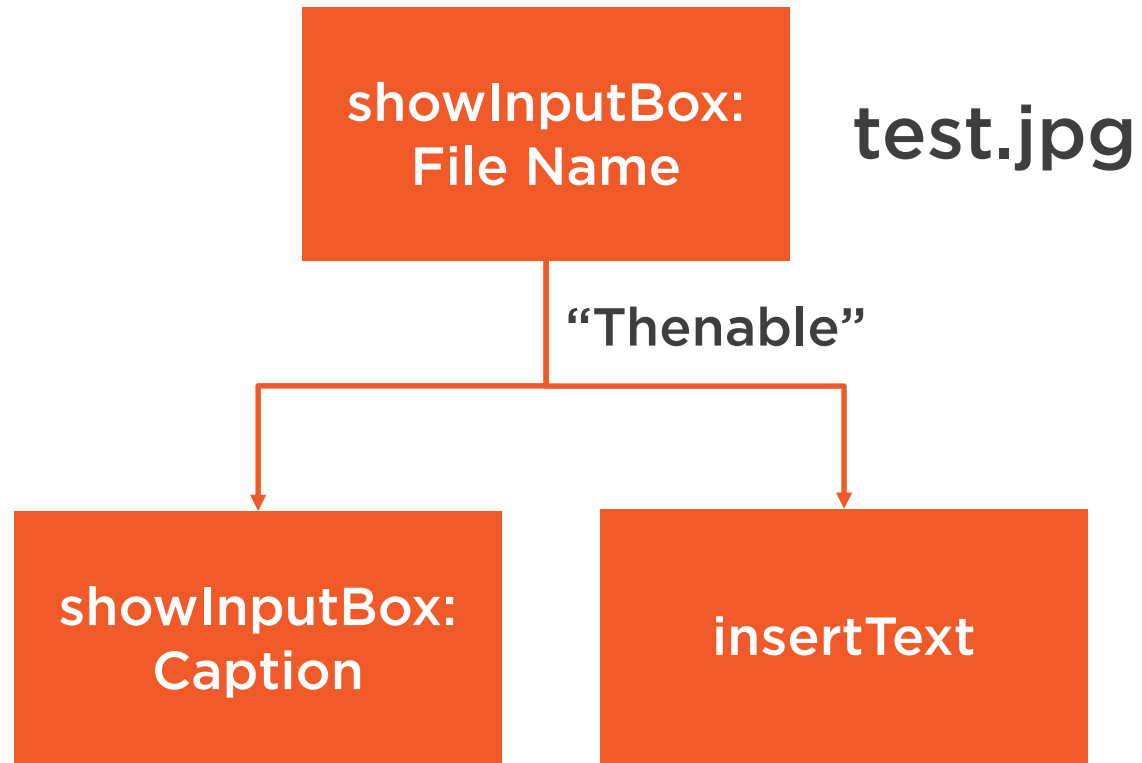
“Thenable”

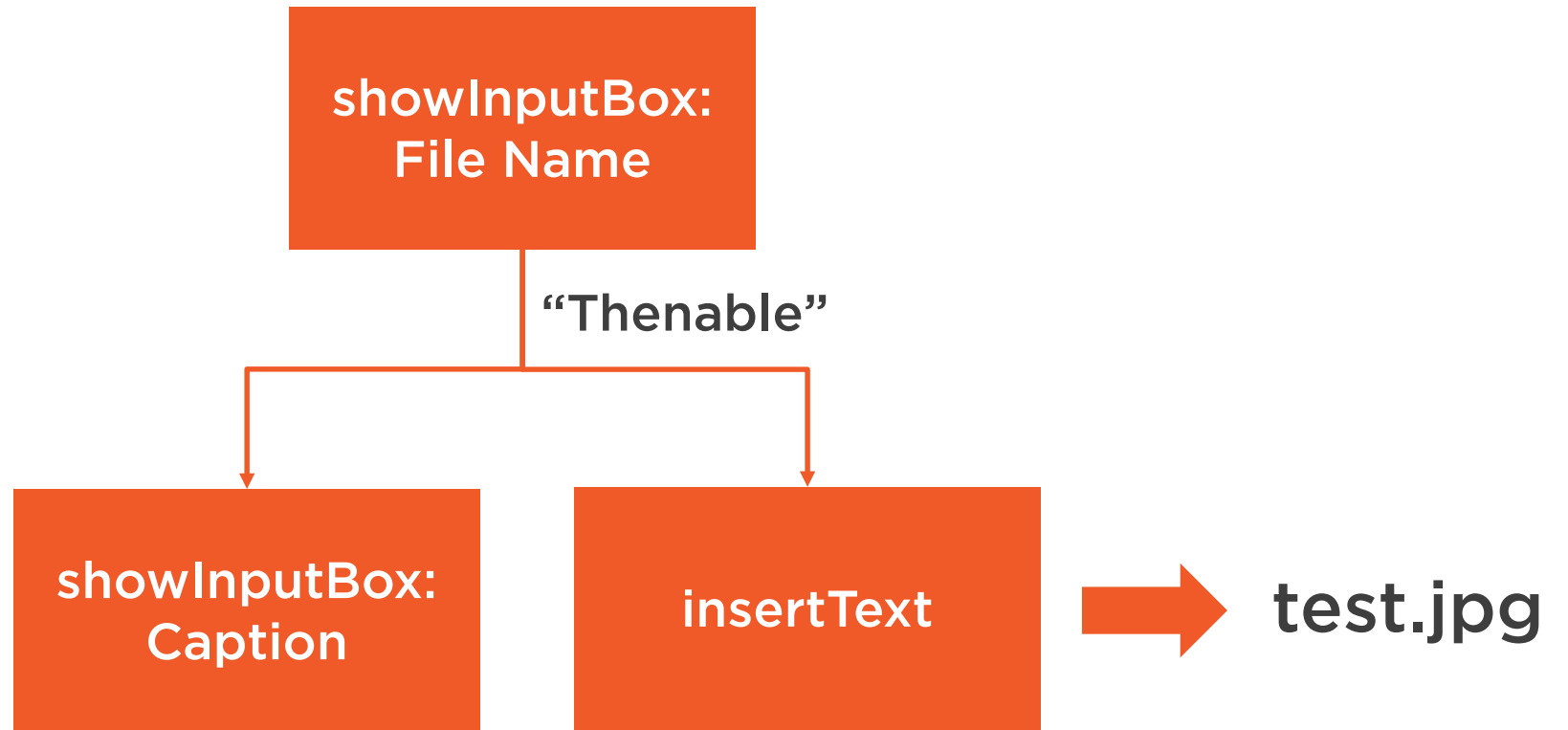
insertText

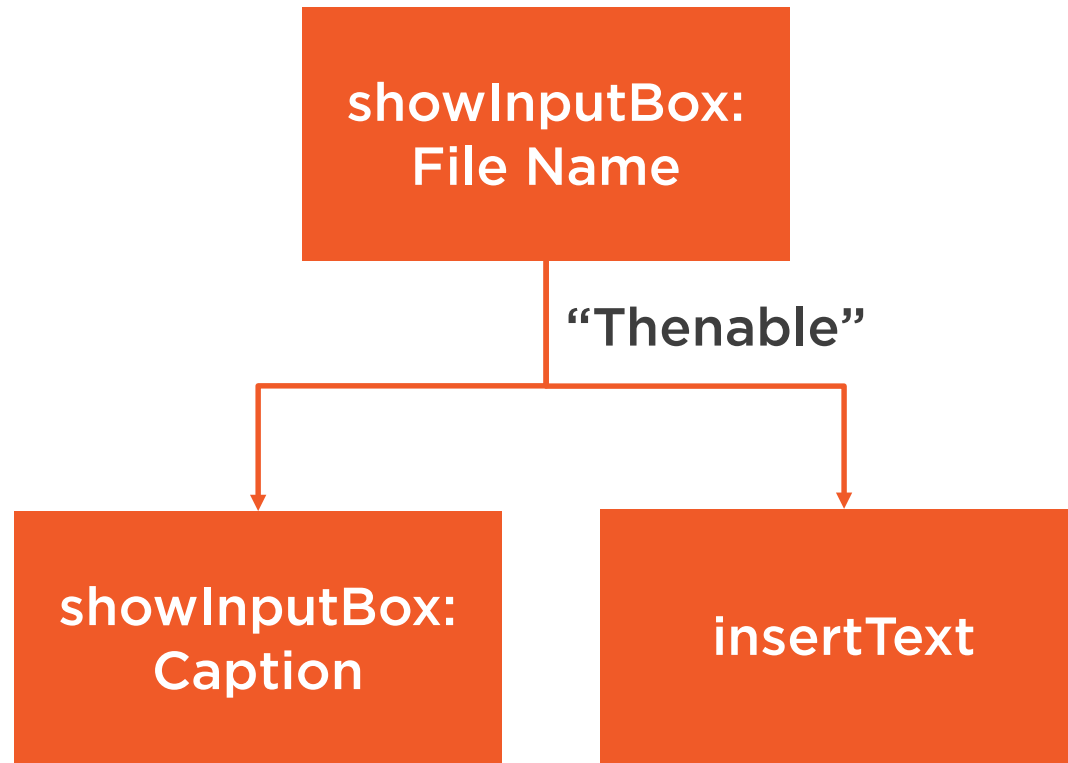
```
graph TD; A[showInputDialog: File Name] -- "Thenable" --> B[showInputDialog: Caption]; B -- "Thenable" --> C[insertText];
```


showInputBox:
File Name

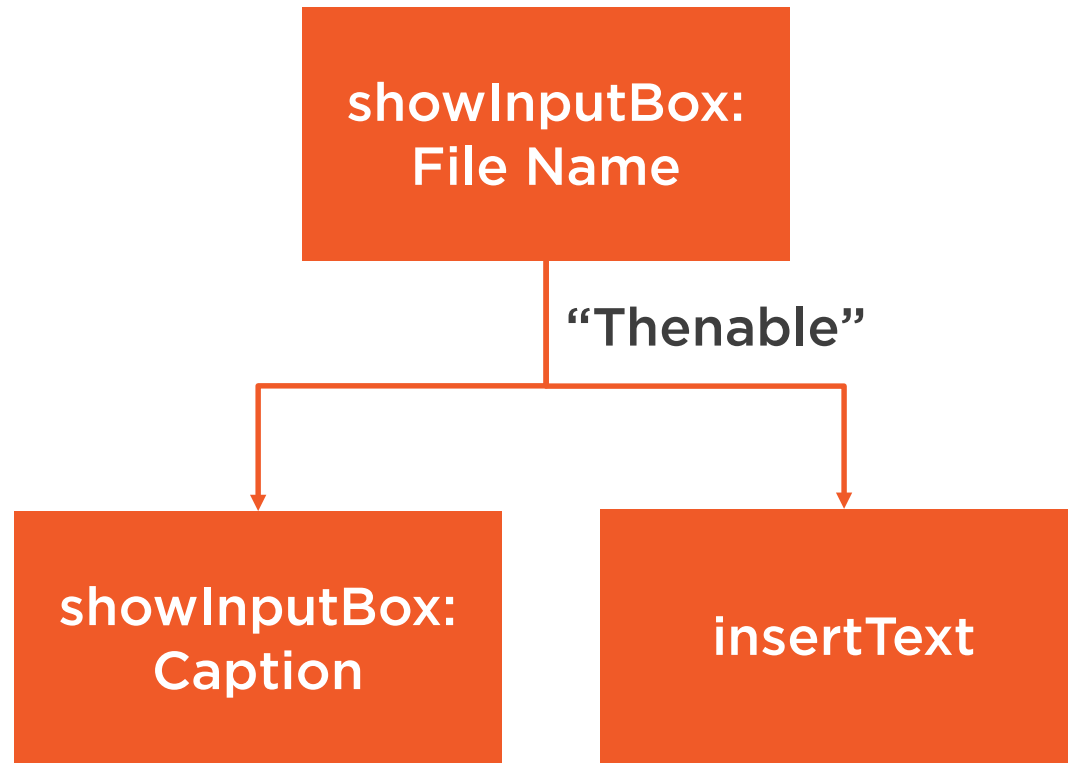








Test Caption



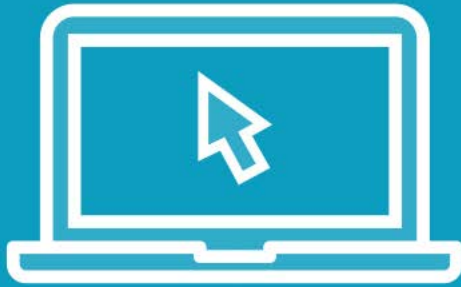






Install Our Extension Locally

Demo



Install Our Extension Locally

What's the Problem?

Create the Project

Define Our Functions

Insert File or Image Links

Write Unit Tests

Interactively Build and Insert Figure Tag

Install Our Extension Locally



Up Next:

Distributing Our Extension