VS Code is designed around an extension model. TypeScript provides a server called TSServer that provides information which supports quick-info, completions, etc., then VS Code acts as a client which queries the server when this information is needed.

For example, VS Code queries TSServer for quick-info when the user's mouse hovers over a variable by sending a message to TSServer. TSServer will respond with information such as the appropriate type, and the styling to apply to the text that describes the type.

Organizationally, the client-side code for communicating with the TypeScript server lives in `extensions/typescript-language-features` in the VS Code repository.[1]

Meanwhile, the server-side code lives in `src/services` and `src/server` of the TypeScript repository.

Using stable VS Code to Debug Stable TSServer

There are two steps to this:

- Launch VS Code with an extra environment variable, and different user profile.
- Connect to this VS Code's TSServer.

To launch VS Code with a different profile and a debug copy of TSServer:

```
# Sets the TSServer port to 5667, this can be any number
# Sets the user-data directory to be ~/.vscode-debug/ instead of ~/.vscode/

TSS_DEBUG=5667 code --user-data-dir ~/.vscode-debug/
```

This will open VS Code as a separate app from your current one, it may have some of your extensions but not your settings. As long as you consistently use the above command, then you can save settings for debugging between sessions.

Optionally you can use `TSS_DEBUG_BRK` to have the TSServer wait for your debugger before launching.

This will launch a debug TSServer which you can connect to from inside the TypeScript codebase. Open up the TypeScript codebase, and look at the debugging panel. At the top, look to see if there is a drop-down item for debugging by Attaching to VS Code TSServer then select that.

If there isn't, copy the template of `.vscode/launch.template.json` to `.vscode/launch.json` and it should show up.

Select the "Attach by ..." option in the dropdown for debugging and hit the play button, it will ask you to choose a node instance to connect to. In the above example we used the port 5667, look for that and select it.

That should have you connected to the TSServer for the debugging app version of VS Code while you work in the production version.

Using Stable VS Code with Development TSServer

VS Code chooses where to launch TSServer from via the setting: `typescript.tsdk`. Continuing from above, if you want to have your TSServer use a local copy of TypeScript then change this setting to:

```
{
    "typescript.tsdk": "/path/to/repo/TypeScript/built/local"
```

```
}
```

This is probably enough for most contributors, but if you are doing heavy duty VS Code and TypeScript work, you may want to carry on.

Development VS Code with Development TSServer

We will use a stable version of VS Code to debug a development version of VS Code running against a development version of TSServer.

1. Download/install a stable version of vs code.
2. Follow the instructions to setup a development version of VS Code.[2]
3. Clone the TypeScript repo locally, and follow the instructions on building TypeScript.
4. Update the User Settings in the development version of VS Code, to point to the `built/local` directory of your local TypeScript repository.

   This will look something like the following:

   ```
   {
      "typescript.tsdk": "/path/to/repo/TypeScript/built/local"
   }
   ```

   You may instead update this in the Workspace Settings for a project as well, but you will have to remember that the development version of TSServer will only be in effect within that project.

From here, there are different steps for debugging the client- and server-side, respectively.

Debugging tsserver (server-side)

1. Choose an available port to debug TSServer using either of the following two methods (in the rest of this guide, we assume you chose 5859):

   - In a shell, export the `TSS_DEBUG` environment variable to an open port. We will run the development VS Code instance from within that shell.

   For most Unix-like shells (e.g. bash), this will be something like

   ```
   export TSS_DEBUG=5859
   ```

   For PowerShell, this is something like

   ```
   $env:TSS_DEBUG = 5859
   ```

   - Alternatively, manually edit `extensions/typescript/src/typescriptServiceClient.ts` in your development-side VS Code, setting the port to an open one.

2. Update `launch.json` with an option to attach to the node instance, with sourcemaps from your `built/local` folder.

For VS Code v1.13.1+ and Node v8.0+, your `launch.json` might look like the following:

```
{
    "version": "0.2.0",
    "configurations": [
        // Other configs
        {
            "name": "Attach to TS Server",
            "type": "node",
            "request": "attach",
            "protocol": "inspector",
            "port": 5859,
            "sourceMaps": true,
            "outFiles": ["/path/to/repo/TypeScript/built/local"],
        }
    ]
}
```

For the same versions of Code, but older versions of Node (e.g. 6.x), you'll need to set `"protocol"` to be `"legacy"`.

3. Launch an instance of your development VS Code, and open a TypeScript file.
4. Launch your stable-release version of VS Code.
5. Attach the stable VS Code instance to the development instance.

Debugging the Extension Host (client-side)

3. Launch an instance of development vs code.
4. Launch an instance of stable vs code.
5. Attach the stable vs code instance to the development instance.

---

[1] In particular, the built-in extension spawns the node instance that loads tsserver via the call to electron.fork() in `extensions/typescript/src/typescriptServiceClient.ts`.

[2] If you are on Linux, be sure to increase the number of file watchers per the fix for ENOSPC errors. for opening medium-large projects like Typescript, the default limit of 8192 is almost certainly too small.