TEST CODE REVIEW

Aditya Jain

B.Tech. Computer Science And Engineering

17th July 2018

1 Message Connections

The whole application uses following types of the messaging connections :

- Between Background Scripts and Content Scripts of the extension
- Between the extension and local application using local server (currently on port *8080*)
- local application and the language server using local server (currently on the port *3000*)

Its easier to visualize using the following diagram:

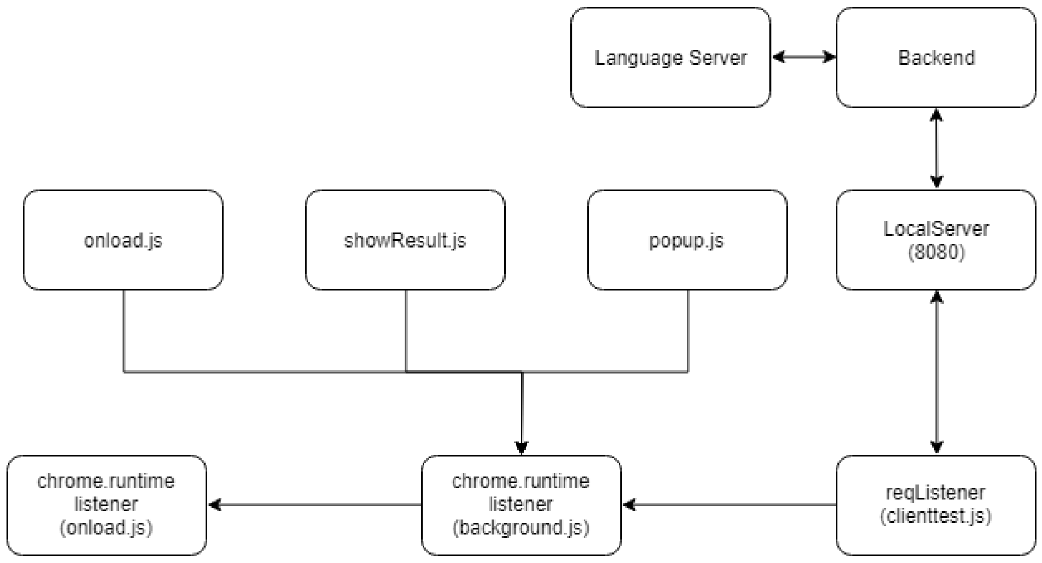
Language Server
Backend
onload.js
showResult.js
popup.js
LocalServer (8080)
chrome.runtime listener (onload.js)
chrome.runtime listener (background.js)
reqListener (clienttest.js)

Figure 1: Connections

We can classify the files in the following manner:

- Background scripts: background.js .
- Content scripts: onload.js,jumble.js, toggle.js, showResult.js, popup.js, clienttest.js
- Local application: start.ts,shellFunctions.ts,myclient.ts,runShell*.

2 Object Specifications

2.1 local server connection objects

2.1.1 From local server to extension

- Sent from *handleRequestBlob,handleRequestPull or handleRequestQuery* functions from the start.ts file when the requested file is not present in serverWorking folder

```
{
        method:"repoNotInServerWorking/repoNotInServerWorkingQ
        uery"
}
```

- Sent from *handleRequestBlob,handleRequestPull* funtions after the server is started

```
{
        method:"serverStarted",
        forReference?:boolean
}
```

- The below two are sent from *handleRequestQuery* function as a result for the blob and pull queries respectively

```
{
        method:"blob",
        query:obj.query,
        definition:result,
        same:boolean,
        repo:string,
        branch:string
}
```

```
{
        method:"pull",
        query:obj.query,
        definition:result,
        same:boolean,
        repo:string,
        branchType:string
}
```

- Sent from the *handleRequestQuery* function when the repo is in serverWorking dir but not in serverRepos

```
{
        method:"reloadToStart",
}
```

- Sent from *handleRequestGitClone* function in response to the git clone request and also sends whether the repo is updated or new as an extra information

```
{
        method:"gitCloneResponse",
        type:"updated/new"
}
```

2.1.2 From extension to local server

- This is send to the server whenever a query is made. The queryObject contains the textDocument and position.

```
{
        method:"query"
        query:queryObject
        type:"blob/pull"
        branchType?:"base/head"
        repo:string
        branch:string
}
```

- Sent from *gitCloneFunction* function in the onload.js file to clone the repo as a response to the request from the background script to the content script

```
{
        method:"gitClone"
        url:string
        repo:string
        downloadType:"https/ssh"
}
```

- The below two objects are sent when a new page is loaded from the *gitCloneFunction* function in the onload.js file. Depending on which kind of page is loaded the object is sent.

```
{
        method:"blob"
        repo:string
        branch:string
}
```

```
{
        method:"pull"
        repo:string
        branchBase:string
        branchHead:string
}
```

2.2 Content script to background

- When the extension receives the *serverStarted* object from the local server the below object is sent as to show the notification in the browser

```
{
        method:"showServerNotification"
}
```

- The following object is send when the extension receives it from the local server.

```
{
        method:"gitCloneResponse",
        type:"updated/new"
}
```

- When the *"repoNotInServerWorking"* object is received at the extension it send the same method and the suggested url to clone the repo to the background page. If there is no suggested url then "undefined" is sent

```
{
        method:"repoNotInServerWorking/repoNotInServerWorkingQ
        uery",
        url:string
}
```

- The following object is just passed from the content script (clienttest.js) to the background.js when it is received from the local server

```
{
        method:"reloadToStart",
}
```

- This object is sent when the result of a query is in different file . Presently it is sent from *showDiffBlobResult* function in the *showResult.js* file. It contains the url of the new tab to be created.

```
{
        method:"openNewTab",
        newLocation:string
}
```

- Sent from the *popupCloneFunction* from the popup.js file when the button in the browserAction window is clicked. The type is dependent on which button is clicked.This is the first request in the gitClone process.

```
{
        method:"gitCloneRequestFromPopup",
        type:"https/ssh"
}
```

- When the local server is not running this is sent as to show the server not running notification in the browser . It is sent from the *sendToServer* function in the onload.js file

```
{
        method:"serverNotRunning",
}
```

- Sent from the *gitCloneFunction* in the onload.js file when it receives the "gitClone" request from the background and after sending the request to the local server.

```
{
        method:"sentForCloning",
        repo:string
}
```

- If the page is not valid for cloning the following object is sent from *gitCloneFunction* to show the info in the browser

```
{
        method:"notValidClonePage",
        url:string
}
```

2.3 Background to ContentScripts

The only message that the Bakground script sends to the content script is when the popup button in the browserAction is clicked. The button sends a message to the background.js about the request which the background page forwards to the content script.

```
{
        method:”gitClone”,
        type:”https/ssh”
}
```