Debugger

When developing a Wakanda application, you will often need to trace the JavaScript code that is executed on both the client and the server.

On the client-side, you can use any debugging tool in your Web browser that you are comfortable with, e.g., Firebug or Google Chrome debugger.

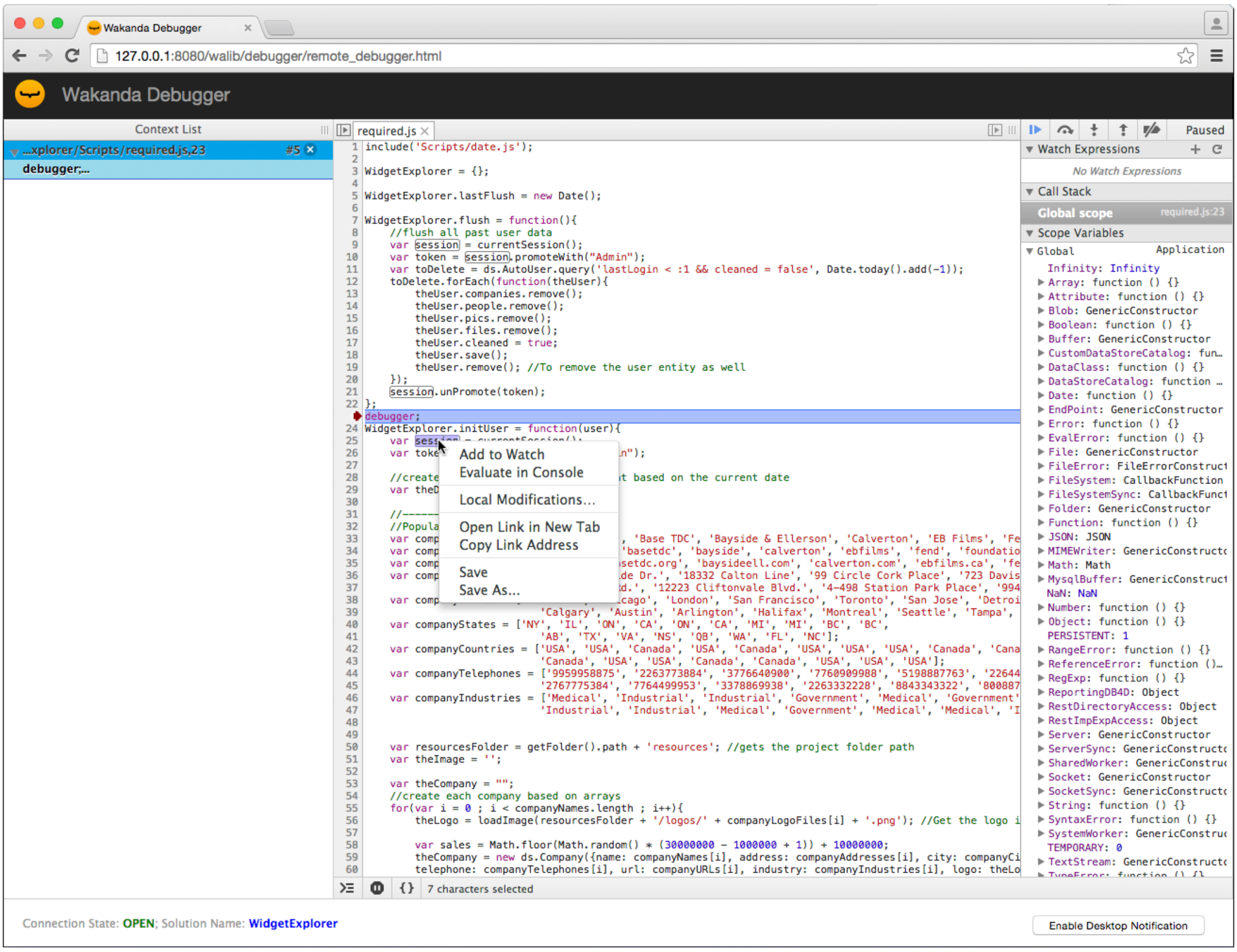
On the server-side, Wakanda allows you to choose between two debugger options:

- **No Debugger (performance)**: This option allows you to turn off the debugger completely. With this option selected, the debugger is not opened for any reason (error in the code, a breakpoint, or a call to the debugger function) and the performance of your solution is optimized.
- **Use Debugger**: This option activates the **Wakanda Debugger**, a standard JavaScript debugger available through Google Chrome and Safari browsers (documented in the *Wakanda Server Reference Guide*).

Wakanda's Web Debugger is automatically launched with Wakanda Server. You can view the status of the Debugger in Wakanda Studio's footer. If you try to run a JavaScript file with the server off, the Debugger will be launched.

Wakanda Debugger

The Wakanda Debugger opens in your default browser window:



```
include('Scripts/date.js');

WidgetExplorer = {};

WidgetExplorer.lastFlush = new Date();

WidgetExplorer.flush = function(){
    //flush all past user data
    var session = currentSession();
    var token = session.promoteWith("Admin");
    var toDelete = ds.AutoUser.query('lastLogin < :1 && cleaned = false', Date.today().add(-1));
    toDelete.forEach(function(theUser){
        theUser.companies.remove();
        theUser.people.remove();
        theUser.pics.remove();
        theUser.files.remove();
        theUser.cleaned = true;
        theUser.save();
        theUser.remove(); //To remove the user entity as well
    });
    session.unPromote(token);
};
debugger;
WidgetExplorer.initUser = function(user){
    var session = currentSession();
    var toke                                n");

    //create                                t based on the current date
    var theD

    //------
    //Popula
    var comp                               , 'Base TDC', 'Bayside & Ellerson', 'Calverton', 'EB Films', 'Fe
    var comp                               'basetdc', 'bayside', 'calverton', 'ebfilms', 'fend', 'foundatio
    var comp                               setdc.org', 'baysideell.com', 'calverton.com', 'ebfilms.ca', 'fe
    var comp                               de Dr.', '18332 Calton Line', '99 Circle Cork Place', '723 Davis
                                           d.', '12223 Cliftonvale Blvd.', '4-498 Station Park Place', '994
    var comp                               cago', 'London', 'San Francisco', 'Toronto', 'San Jose', 'Detroi
                            'Calgary', 'Austin', 'Arlington', 'Halifax', 'Montreal', 'Seattle', 'Tampa',
    var companyStates = ['NY', 'IL', 'ON', 'CA', 'ON', 'CA', 'MI', 'MI', 'BC', 'BC',
                            'AB', 'TX', 'VA', 'NS', 'QB', 'WA', 'FL', 'NC'];
    var companyCountries = ['USA', 'USA', 'Canada', 'USA', 'Canada', 'USA', 'USA', 'USA', 'Canada', 'Cana
                            'Canada', 'USA', 'USA', 'Canada', 'Canada', 'USA', 'USA', 'USA'];
    var companyTelephones = ['9959958875', '2263773884', '3776640900', '7760909988', '5198887763', '22644
                            '2767775384', '7764499953', '3378869938', '2263332228', '8843343322', '800887
    var companyIndustries = ['Medical', 'Industrial', 'Industrial', 'Government', 'Medical', 'Government'
                            'Industrial', 'Industrial', 'Medical', 'Government', 'Medical', 'Medical', 'I


    var resourcesFolder = getFolder().path + 'resources'; //gets the project folder path
    var theImage = '';

    var theCompany = "";
    //create each company based on arrays
    for(var i = 0 ; i < companyNames.length ; i++){
        theLogo = loadImage(resourcesFolder + '/logos/' + companyLogoFiles[i] + '.png'); //Get the logo i

        var sales = Math.floor(Math.random() * (30000000 - 1000000 + 1)) + 10000000;
        theCompany = new ds.Company({name: companyNames[i], address: companyAddresses[i], city: companyCi
        telephone: companyTelephones[i], url: companyURLs[i], industry: companyIndustries[i], logo: theLo
```

```
▼ Global                                  Application
    Infinity: Infinity
  ▶ Array: function () {}
  ▶ Attribute: function () {}
  ▶ Blob: GenericConstructor
  ▶ Boolean: function () {}
  ▶ Buffer: GenericConstructor
  ▶ CustomDataStoreCatalog: fun…
  ▶ DataClass: function () {}
  ▶ DataStoreCatalog: function …
  ▶ Date: function () {}
  ▶ EndPoint: GenericConstructor
  ▶ Error: function () {}
  ▶ EvalError: function () {}
  ▶ File: GenericConstructor
  ▶ FileError: FileErrorConstruct
  ▶ FileSystem: CallbackFunction
  ▶ FileSystemSync: CallbackFunct
  ▶ Folder: GenericConstructor
  ▶ Function: function () {}
  ▶ JSON: JSON
  ▶ MIMEWriter: GenericConstructo
  ▶ Math: Math
  ▶ MysqlBuffer: GenericConstruct
    NaN: NaN
  ▶ Number: function () {}
  ▶ Object: function () {}
    PERSISTENT: 1
  ▶ RangeError: function () {}
  ▶ ReferenceError: function ()…
  ▶ RegExp: function () {}
  ▶ ReportingDB4D: Object
  ▶ RestDirectoryAccess: Object
  ▶ RestImpExpAccess: Object
  ▶ Server: GenericConstructor
  ▶ ServerSync: GenericConstructo
  ▶ SharedWorker: GenericConstruc
  ▶ Socket: GenericConstructor
  ▶ SocketSync: GenericConstructo
  ▶ String: function () {}
  ▶ SyntaxError: function () {}
  ▶ SystemWorker: GenericConstruc
    TEMPORARY: 0
  ▶ TextStream: GenericConstructo
```

Connection State: OPEN; Solution Name: WidgetExplorer

For information on how to use it, refer to **Working in a Debugger Window**.

Using a command line

You can select or deactivate the Debugger at startup through a command line. For more information, refer to the **Administrating Wakanda Server (Unix)** or **Administrating Wakanda Server (Windows)** section.

Accessing the Debugger

If you have set up your solution's **Admin Access Control**, you will be prompted to enter your name and password when connecting to the debugger. Only users belonging to the "Admin" group will be allowed to access the Debugger. For more information, refer to the **Debugger Access** chapter in the **Data Security and Access Control** manual.

Debugging on a Remote Server

Besides debugging your project on a local Wakanda Server, you can also debug your server-side JavaScript files on a remote server. The local project and the remote project must be identical because when you choose to run the local JavaScript file, the version of the file with the same name is executed remotely.

To debug on a remote server from Wakanda Studio, select **Connect to Remote Server** from the **Run** menu. Select the server containing the same project (running on the same port) or enter in the address of a Wakanda Server running the same project, and click **Connect**.

You can also debug remotely by clicking on the **Debug** button from a remote **Wakanda Server Administration** page.

Enabling HTTPS connections with the Debugger

When you open a debug session with the **Debugger** on a remote machine, the connection automatically uses the https protocol. It means that you need a valid SSL/TLS certificate installed in your solution to allow https connections. Otherwise, the connection will not be allowed and you will get a connection error such as "*Sendind command to Wakanda server has failed*".

Implementation Note: *In previous releases of Wakanda, a self-signed certificate was provided to facilitate remote debugging between two machines. For security reasons, this certificate is not provided anymore, you need to install and use your own certificate.*

For information about how to install a SSL/TLS certificate, please refer to the **Configuring secure connections (SSL/TLS)** section.

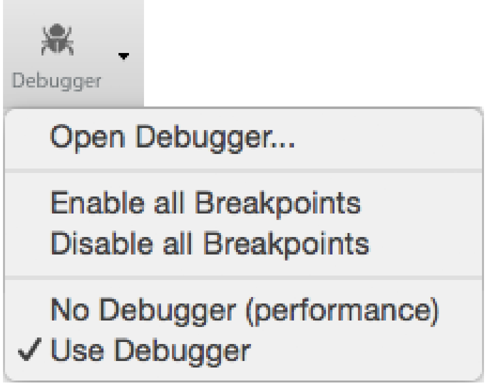
Configuring the Debugger

Enabling the Debugger

By default, Wakanda solutions are configured to work with the **Wakanda Debugger** in your browser. You can disable it by selecting **No Debugger**.

Using the Debugger

You select which debugger to use (or no debugger) for your solution by clicking on the **Debugger** toolbar button menu:



The check mark indicates which option is selected for your solution:

- **No Debugger (performance):** This option allows you to turn off the debugger completely. With this option selected, the debugger is not opened for any reason (error in the code, a breakpoint, or a call to the debugger function) and the performance of your solution is optimized.
- **Use Debugger**: This option launches the Wakanda Debugger in your browser (by default)

Using a command line

You can select or deactivate the Debugger at startup through a command line. For more information, refer to the **Administrating Wakanda Server (Unix)** or **Administrating Wakanda Server (Windows)** section.

Accessing the Debugger

If you have set up your solution's **Admin Access Control**, you will be prompted to enter your name and password when connecting to the debugger. Only users belonging to the "Admin" group will be allowed to access the Debugger. For more information, refer to the **Debugger Access** chapter in the **Data Security and Access Control** manual.

Debugging on a Remote Server

Besides debugging your project on a local Wakanda Server, you can also debug your server-side JavaScript files on a remote server. The local project and the remote project must be identical because when you choose to run the local JavaScript file, the version of the file with the same name is executed remotely.

To debug on a remote server from Wakanda Studio, select **Connect to Remote Server** from the **Run** menu. Select the server containing the same project (running on the same port) or enter in the address of a Wakanda Server running the same project, and click **Connect**.

You can also debug remotely by clicking on the **Debug** button from a remote **Wakanda Server Administration** page.

Enabling HTTPS connections with the Debugger

When you open a debug session with the **Debugger** on a remote machine, the connection automatically uses the https protocol. It means that you need a valid SSL/TLS certificate installed in your solution to allow https connections. Otherwise, the connection will not be allowed and you will get a connection error such as "*Sendind command to Wakanda server has failed*".

Implementation Note: *In previous releases of Wakanda, a self-signed certificate was provided to facilitate remote debugging between two machines. For security reasons, this certificate is not provided anymore, you need to install and use your own certificate.*

For information about how to install a SSL/TLS certificate, please refer to the **Configuring secure connections (SSL/TLS)** section.

Wakanda Studio Debugger

Calling the Debugger

When the Wakanda Debugger is selected, it is launched in any of the following circumstances:

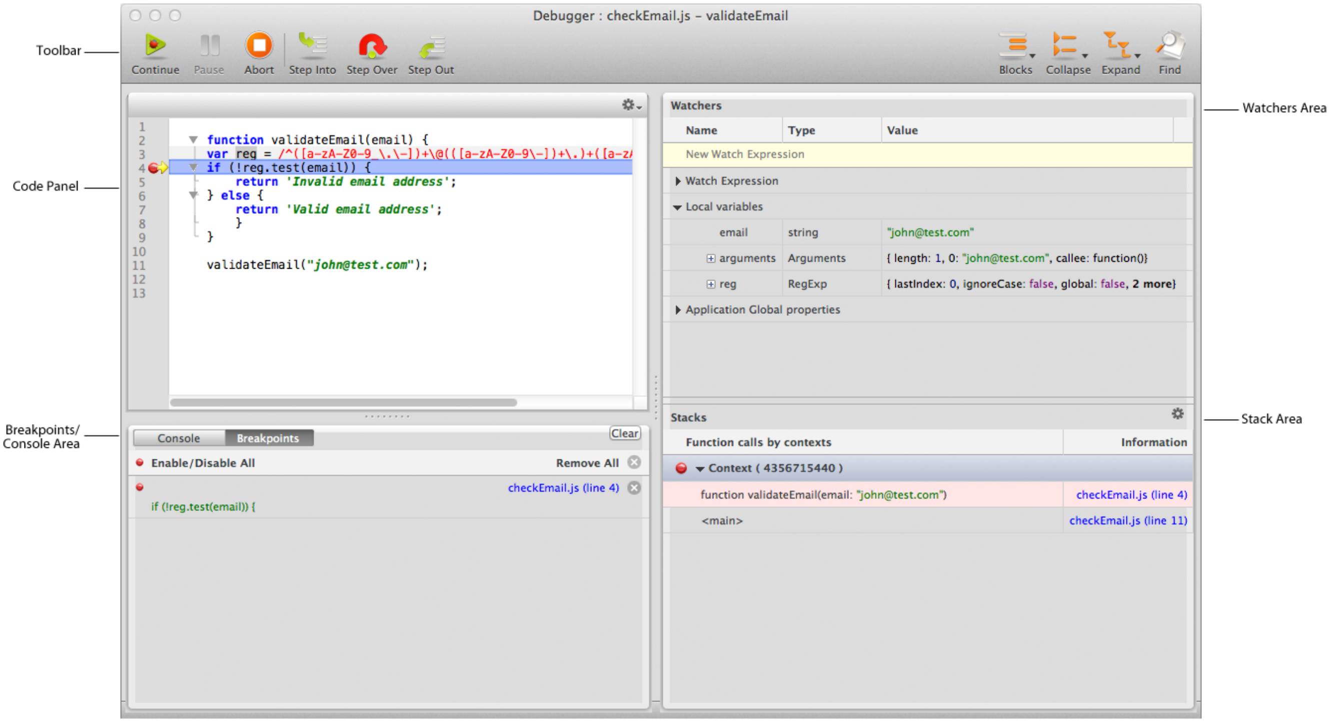
- When you select the **Open Debugger...** option in the Debugger button menu of the Wakanda Studio toolbar,
- When you run a server-side JavaScript file with one or more breakpoints defined,
- When you run a server-side JavaScript file that has an error in it or that provokes an exception,
- When the following line is executed in your server-side JavaScript file:

```
debugger; //opens the debugger
```

Overview

The Wakanda Debugger is made up of the following areas:

- Toolbar
- Code Panel
- Breakpoints/Console Area
- Watchers Area
- Stacks Area



```
function validateEmail(email) {
var reg = /^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA
if (!reg.test(email)) {
    return 'Invalid email address';
} else {
    return 'Valid email address';
    }
}

validateEmail("john@test.com");
```

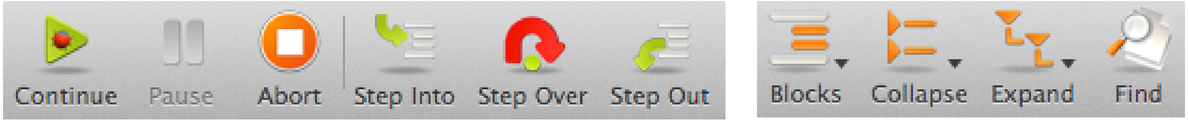
● checkEmail.js (line 4)
if (!reg.test(email)) {

Name	Type	Value
New Watch Expression		
▸ Watch Expression		
▾ Local variables		
email	string	"john@test.com"
⊞ arguments	Arguments	{ length: 1, 0: "john@test.com", callee: function()}
⊞ reg	RegExp	{ lastIndex: 0, ignoreCase: false, global: false, 2 more}
▸ Application Global properties		

Function calls by contexts	Information
● ▾ Context (4356715440)	
function validateEmail(email: "john@test.com")	checkEmail.js (line 4)
<main>	checkEmail.js (line 11)

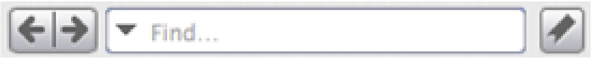
Toolbar

The Code Editor toolbar allows quick access to many key functions that you will be using. You can customize the toolbar by selecting which options to display, how to display them (text only, text and icon or icon only), and if you prefer to display small icons or the larger ones.



The toolbar contains the following default functions:

- **Continue**: Continues execution to the end or to the next breakpoint. The skipped statements are executed, but not stepped through.
- **Pause**: Pauses the debugging.
- **Abort**: Stops the debugging.
- **Step Into**: Executes the current statement, then stops at the next statement. If the current statement is a function or script call, the debugger steps into that function or script, otherwise it stops at the next statement.
- **Step Over**: Executes the current statement, then stops at the next statement. If the current statement is a function or script call, the debugger executes the whole function or script and stops at the next statement after the function call.
- **Step Out**: Steps out of the current function and up one level if the function is nested. If in the main body, the script is executed to the end, or to the next breakpoint. The skipped statements are executed, but not stepped through.
- **Blocks**: This button becomes a menu containing the three options: Select Block, Start of Block, and End of Block.
- **Collapse**: Collapses the selection, current level, or all the blocks.
- **Expand**: Expands the selection, current level, or all the blocks.
- **Find**: Displays a Find area above the editable area where you can find a text string in your file.



If all the items in the toolbar cannot be displayed due to lack of space, the » icon appears in the toolbar. When you click on it, the missing toolbar icons appear in a hierarchical menu.

To close the Debugger, click on the Close Window button in the window’s title bar.

Contextual Menu in the Code Panel

The contextual menu in the Code Panel is similar to the editable area in the Code Editor. It offers you all the same options as well as the following two:

- **Goto Definition**: Allows you to go to the definition of a JavaScript symbol either in the same JavaScript file or in another file in the Code Editor.
- **Find References**: Allows you to find the highlighted/selected value in the Code Editor in all the JavaScript, HTML, CSS, and XML files in the project and display them in the **Find Results** dialog.

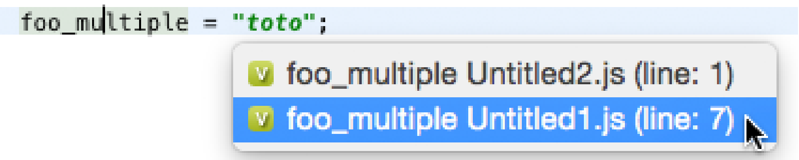
Goto Definition

The **Goto Definition** feature allows you to go directly to the definition of a JavaScript symbol (i.e., a variable, object, array, datastore class, attribute, or method -- anything you defined in Wakanda Studio) in one or more JavaScript files in your current project. You access this feature by either putting your cursor in the name of the JavaScript variable or by highlighting it completely and then you can select **Goto Definition** from the contextual menu.

If your JavaScript variable is named “obj.element1,” you can find its definition either by highlighting it in its entirety or by placing your cursor somewhere in the property “element1.”

If no definition was found, an alert appears.

If one definition was found, your cursor will go to that line either in the same JavaScript file or will open the JavaScript file that contains the JavaScript variable’s definition. If you choose to go to the definition of a datastore class or attribute, the Data Model Editor is opened in a new tab to show you the definition.
If multiple definitions were found in your project, the following pop-up menu appears, allowing you to select which one:



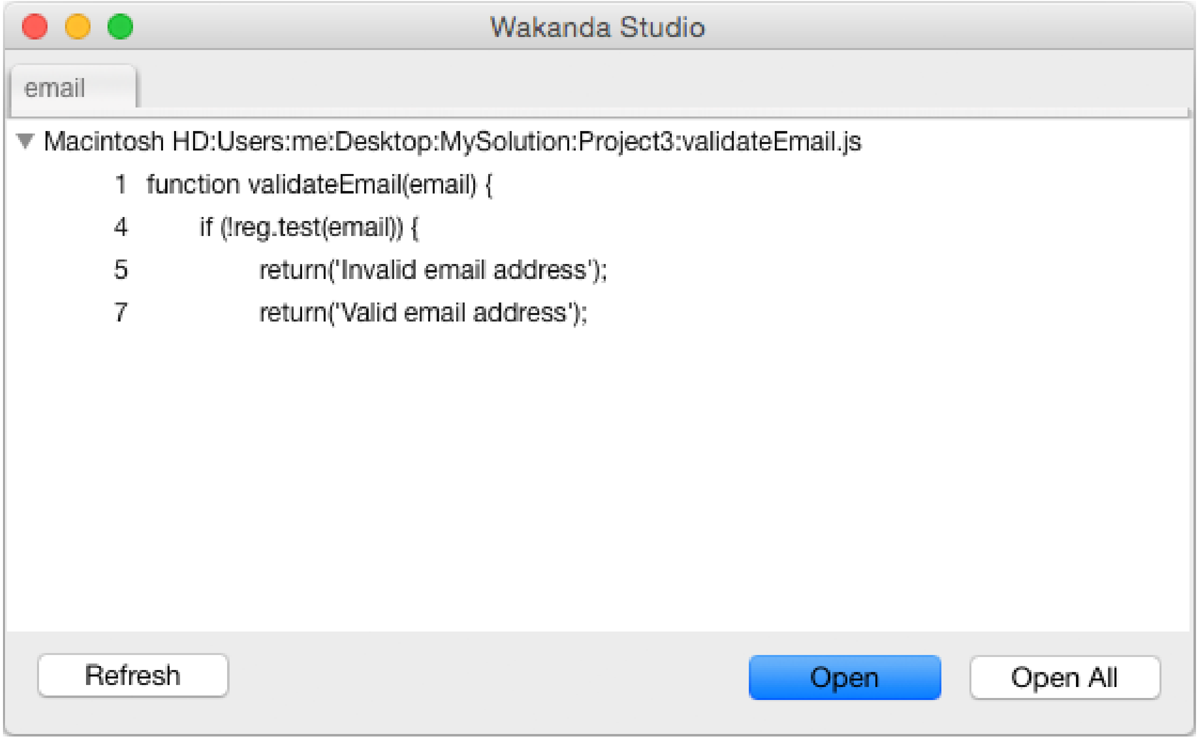
```
foo_multiple = "toto";
```

foo_multiple Untitled2.js (line: 1)
foo_multiple Untitled1.js (line: 7)

__Note__: The icons next to the entry are the same as those in the Outline. For more information, please refer to the Outline section below.

Find References

You can highlight text or a word in the Code Panel and select **Find References** from the contextual menu, Wakanda searches all the JavaScript, HTML, CSS, and XML files in the project and returns the results in the following dialog:



```
▼ Macintosh HD:Users:me:Desktop:MySolution:Project3:validateEmail.js
        1  function validateEmail(email) {
        4      if (!reg.test(email)) {
        5          return('Invalid email address');
        7          return('Valid email address');
```