Approving The JADE Add-in

The JavaScript Automation Development Environment (JADE) add-in for Microsoft Excel allows users to automate their Excel workflow using JavaScript. This guide is to assist the Microsoft Office Store to understand the add-in during the validation process.

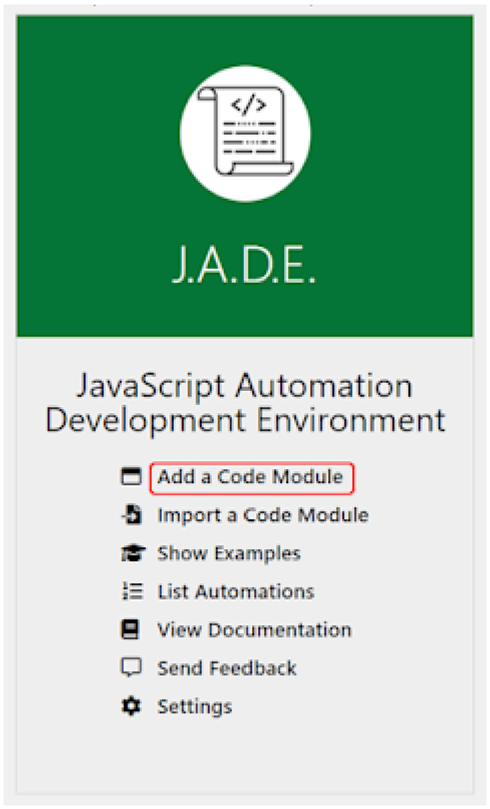
The JADE add-in is fundamentally different from the well-established "Script Lab" add-in for two reasons:

1. JADE targets Excel users, not add-in developers. As such, effort is taken to present a simplified subset of the JavaScript language to make it more accessible to Excel power users, rather than to developers.

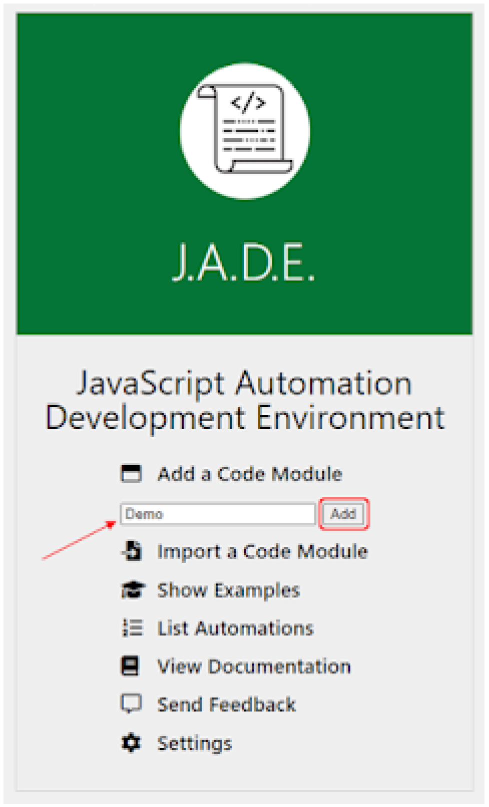
2. The primary place that JADE stores code written by a user is in the Excel workbook, so a workflow automation travels with the workbook.

1. Adding a Code Module and Running a Function

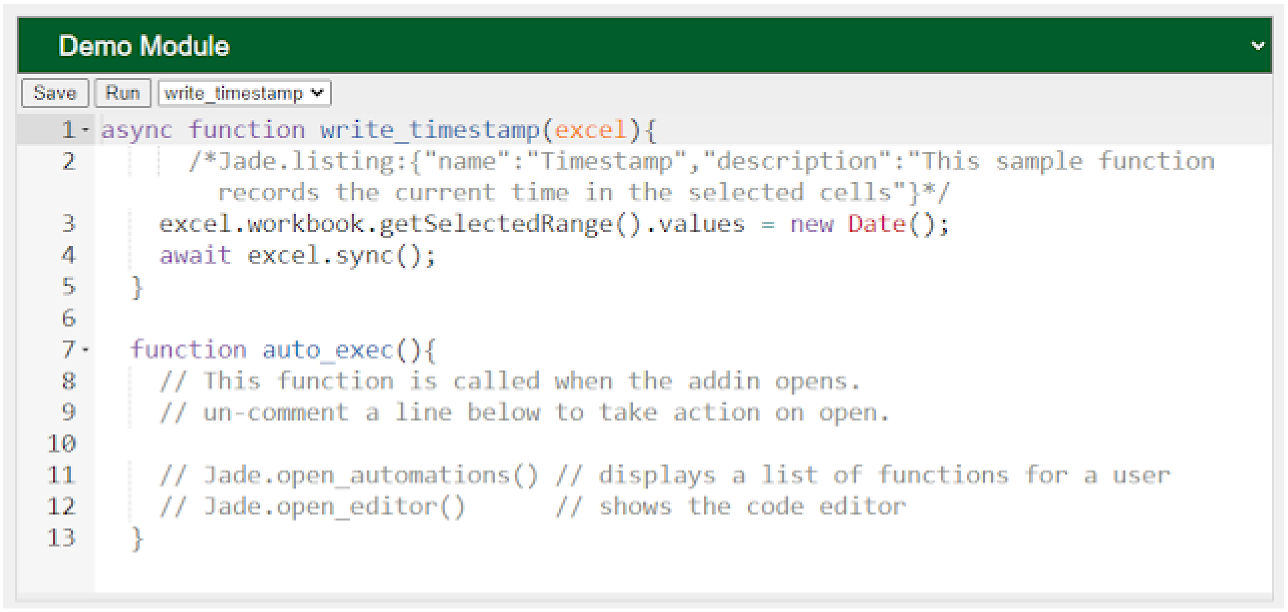
This guide will step you through working with each of the JADE's major features. When the add-in opens you will see the following introductory screen (called the "Home panel"). Let's begin by clicking "Add a Code Module."



Now, enter "Demo" for the module name and click "Add".



This will create the module seen below. You will want to resize the add-in to have a better experience working with the code.



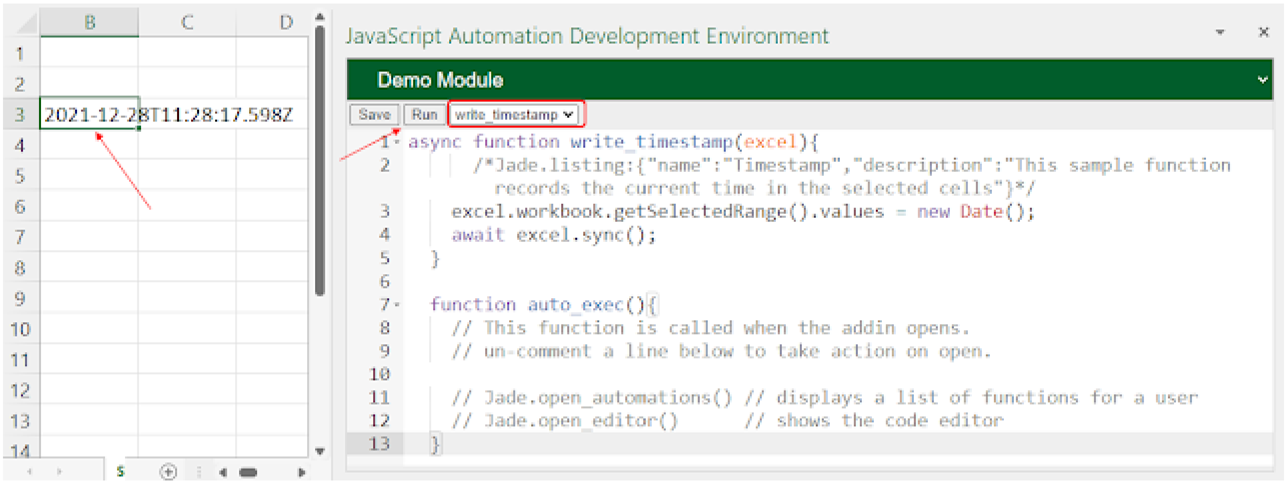
```
async function write_timestamp(excel){
    /*Jade.listing:{"name":"Timestamp","description":"This sample function
      records the current time in the selected cells"}*/
  excel.workbook.getSelectedRange().values = new Date();
  await excel.sync();
}

function auto_exec(){
  // This function is called when the addin opens.
  // un-comment a line below to take action on open.

  // Jade.open_automations() // displays a list of functions for a user
  // Jade.open_editor()      // shows the code editor
}
```

When a new module is created, it begins with the example code you see in the image above. This module comprises the following two functions

1. write_timestamp: A simple function to get the time from the system clock and put it in the selected cell or cells

2. auto_exec: A specially-named function that is called automatically when the add-in loads. When the module is created, the sample code in this function is disabled using the comment symbol ("//")

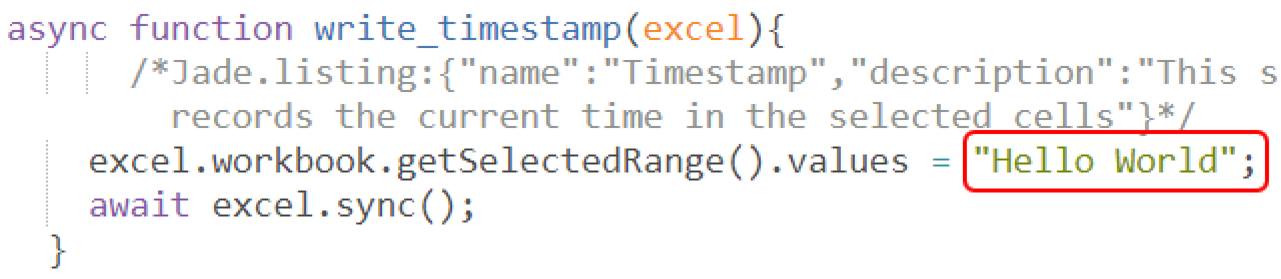
You will notice that write_timestamp is currently shown in the "function name" drop-down selector. Click the "Run" button to execute the function and write the current date and time into the active cell.



2021-12-28T11:28:17.598Z

```
async function write_timestamp(excel){
    /*Jade.listing:{"name":"Timestamp","description":"This sample function
      records the current time in the selected cells"}*/
  excel.workbook.getSelectedRange().values = new Date();
  await excel.sync();
}

function auto_exec(){
  // This function is called when the addin opens.
  // un-comment a line below to take action on open.

  // Jade.open_automations() // displays a list of functions for a user
  // Jade.open_editor()      // shows the code editor
}
```

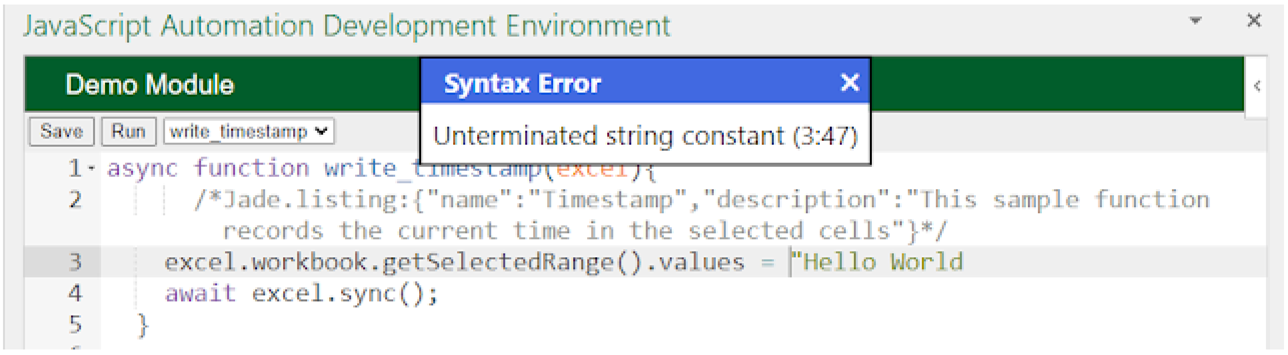
Change the write-timestamp function to write "Hello World" into the selected cell by modifying the code to appear as follows:



```
async function write_timestamp(excel){
    /*Jade.listing:{"name":"Timestamp","description":"This s
      records the current time in the selected cells"}*/
  excel.workbook.getSelectedRange().values = "Hello World";
  await excel.sync();
}
```

Click the "Run" button again to see that the write_timestamp function now writes "Hello World" to the selected cell instead of the current date and time.

If you were to create a syntax error in the code by omitting the closing quote. you would receive an error message telling you where the error occurs as seen here.



Syntax Error

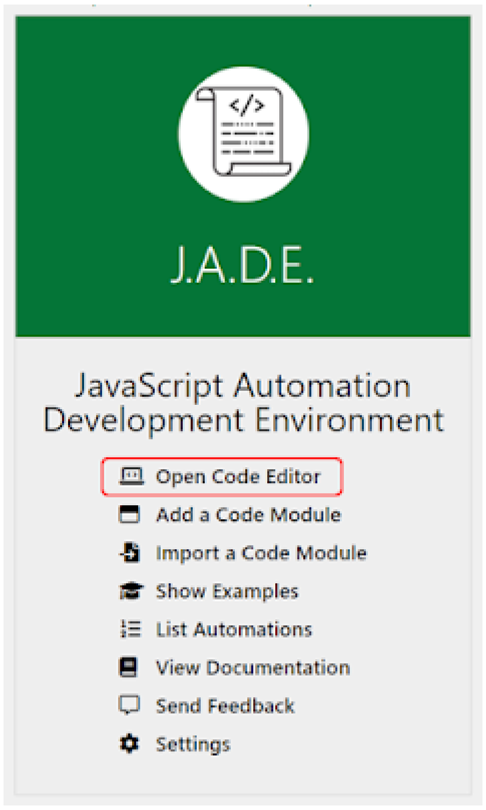
Unterminated string constant (3:47)

```
1  async function write_timestamp(excel){
2      /*Jade.listing:{"name":"Timestamp","description":"This sample function
         records the current time in the selected cells"}*/
3    excel.workbook.getSelectedRange().values = "Hello World
4    await excel.sync();
5  }
```

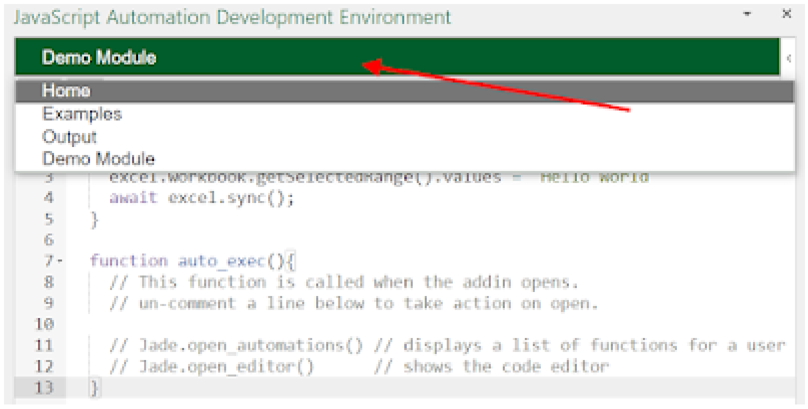
Be sure to correct the syntax error before moving on.

Each time the code is run, the code module is saved to the workbook, so you can now close the add-in without losing your code. You can also save and close the workbook without losing your code because the is a part of the workbook file. To save code changes without running them, just click the "Save" button.

The next time you open the add-in while this workbook is active, you will see a new option to allow you to open the code module that already exists in the workbook, as seen here.



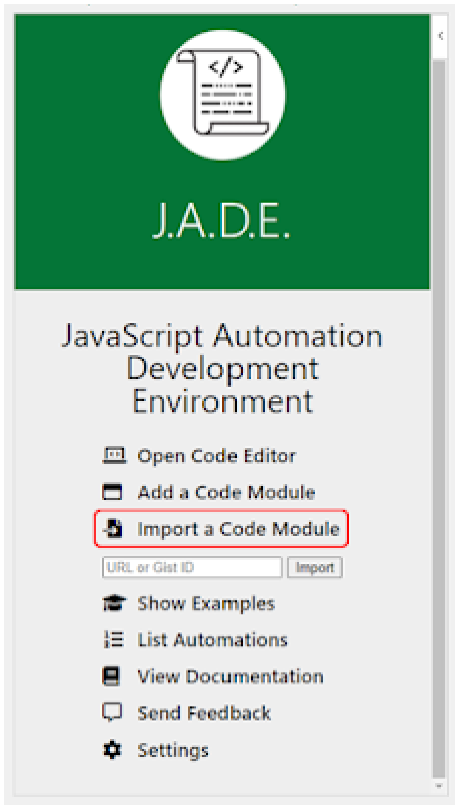
To return to the Home panel from the code editor without closing the add-in, just click the green bar that says "Demo Module" a the top of the add-in pane and choose the "Home" option as seen below.



```
    excel.workbook.getSelectedRange().values = "Hello World"
    await excel.sync();
  }

  function auto_exec(){
    // This function is called when the addin opens.
    // un-comment a line below to take action on open.

    // Jade.open_automations() // displays a list of functions for a user
    // Jade.open_editor()      // shows the code editor
  }
```

2. Import a Code Module

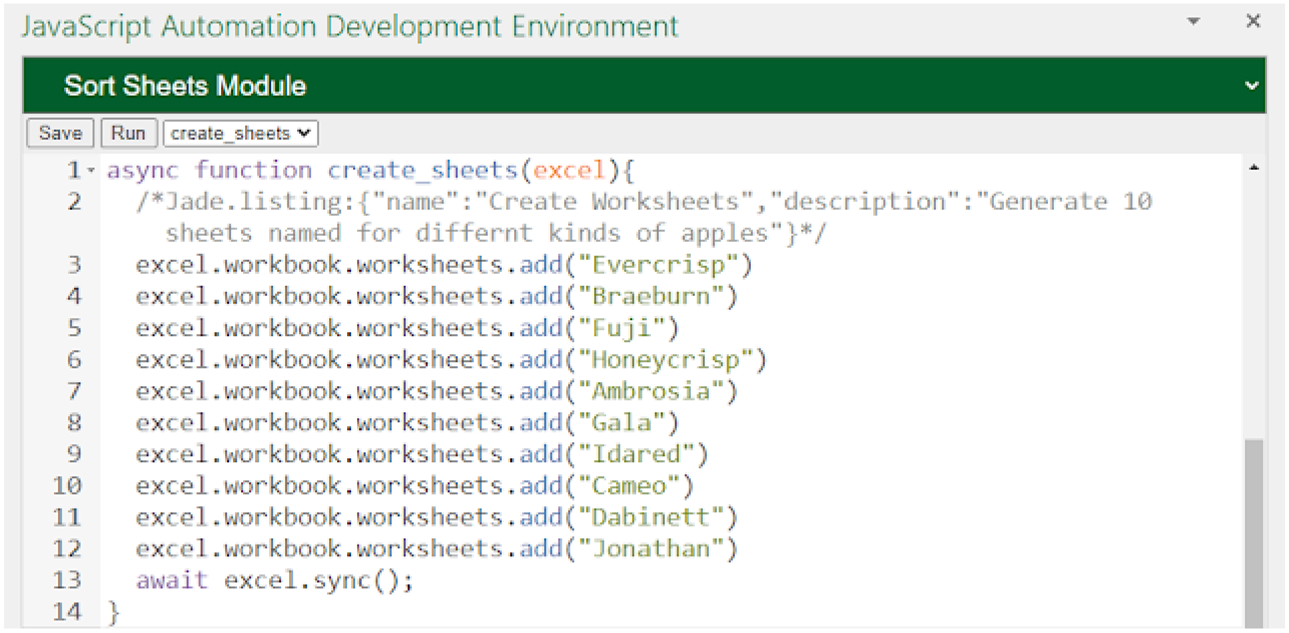
The JADE add-in has the ability to import a code module that has been saved as a Gist on github.com. When you are on the home panel, just click "Import a Code Module" and enter the Gist ID of the gist that contains the code to import.



Enter the following Gist ID to import a module that contains a function to create several worksheets and another to sort the worksheets of a workbook in alphabetical order:

89e47b63aa398a8c915235b7bfda0706

The JADE add-in will display the newly imported module as follows.

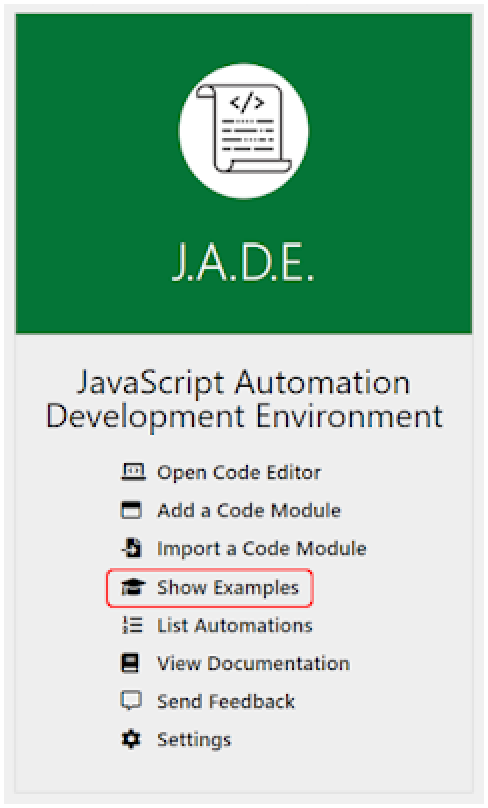


```
async function create_sheets(excel){
  /*Jade.listing:{"name":"Create Worksheets","description":"Generate 10
    sheets named for differnt kinds of apples"}*/
  excel.workbook.worksheets.add("Evercrisp")
  excel.workbook.worksheets.add("Braeburn")
  excel.workbook.worksheets.add("Fuji")
  excel.workbook.worksheets.add("Honeycrisp")
  excel.workbook.worksheets.add("Ambrosia")
  excel.workbook.worksheets.add("Gala")
  excel.workbook.worksheets.add("Idared")
  excel.workbook.worksheets.add("Cameo")
  excel.workbook.worksheets.add("Dabinett")
  excel.workbook.worksheets.add("Jonathan")
  await excel.sync();
}
```

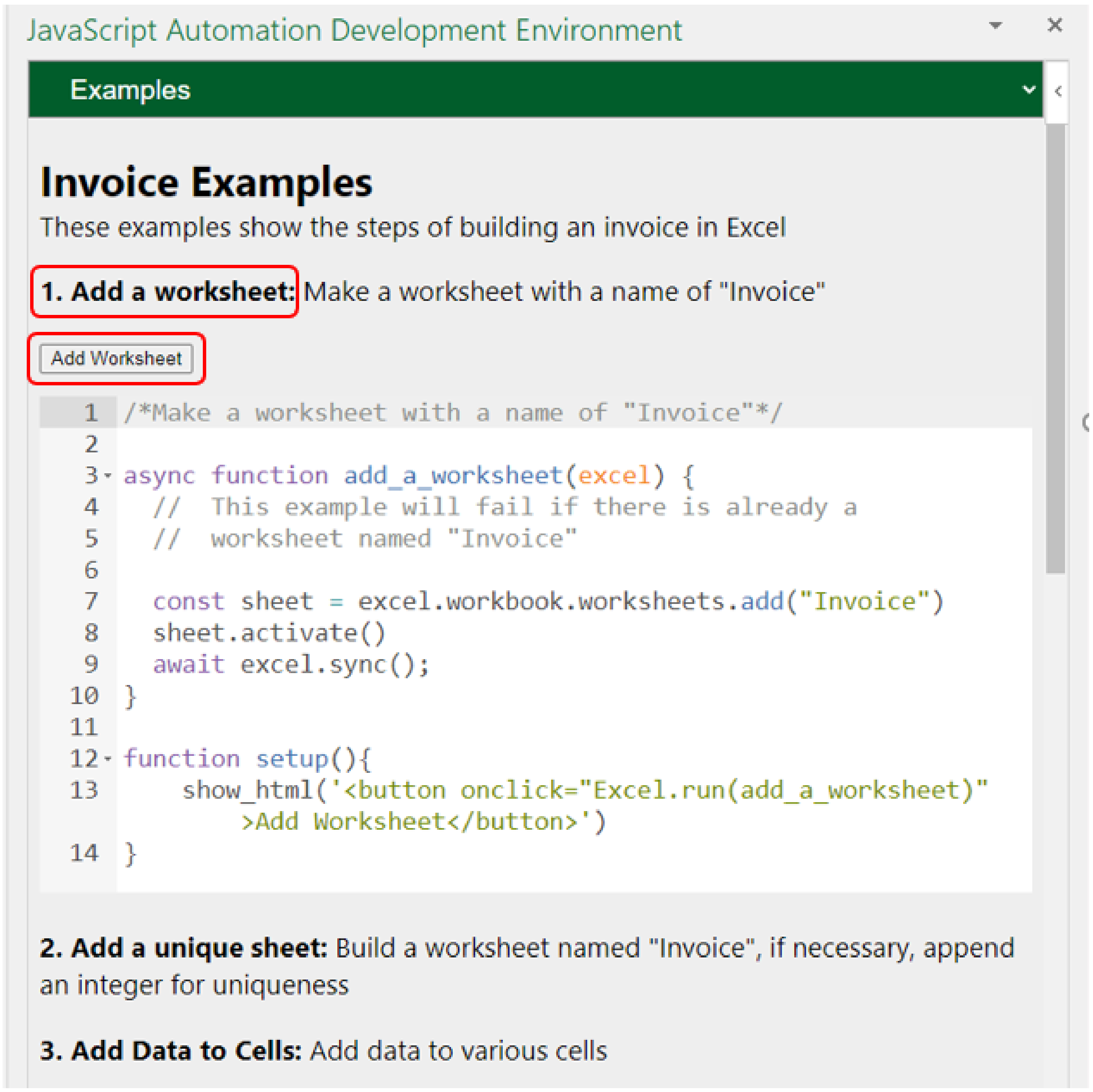
If you like, click the "Run" button to add ten new worksheets, named after ten different varieties of apple. Then change the function-name drop-down to run to the "sort_sheet" function and watch the newly created sheets be rearranged into alphabetical order.

3. Show Examples

On the Home panel of the JADE add-in, you will see an option labeled "Show Examples." Click it to display a list of examples for accomplishing different tasks in a workbook.



On the Examples panel, click the name of the first example, "Add a worksheet" and the example will be displayed. Then, click the button labeled "Add Worksheet" to create and activate a new worksheet named "Invoice." Because the examples are editable, you can change "Invoice" on line 7 and click the "Add Worksheet" button again to create a worksheet with the name you chose.



Invoice Examples

These examples show the steps of building an invoice in Excel

1. Add a worksheet: Make a worksheet with a name of "Invoice"

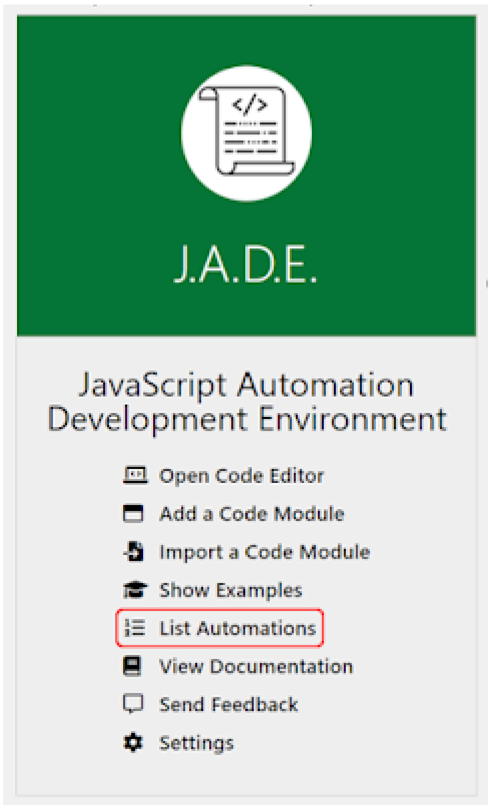
```
/*Make a worksheet with a name of "Invoice"*/

async function add_a_worksheet(excel) {
  //  This example will fail if there is already a
  //  worksheet named "Invoice"

  const sheet = excel.workbook.worksheets.add("Invoice")
  sheet.activate()
  await excel.sync();
}

function setup(){
    show_html('<button onclick="Excel.run(add_a_worksheet)"
        >Add Worksheet</button>')
}
```

2. Add a unique sheet: Build a worksheet named "Invoice", if necessary, append an integer for uniqueness

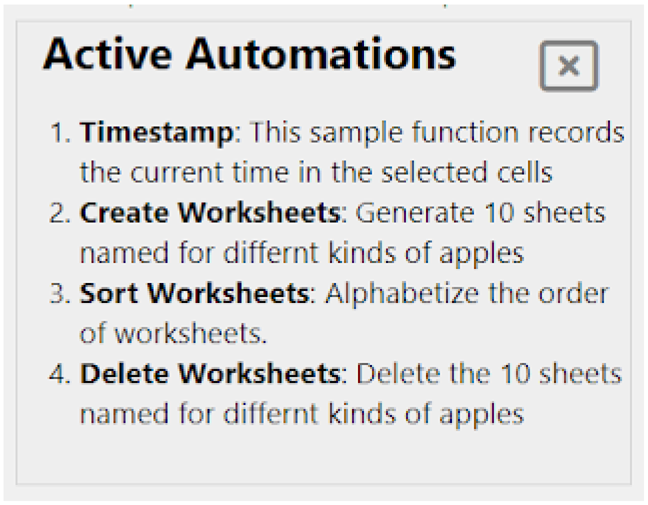
3. Add Data to Cells: Add data to various cells

4. List Automations

On the Home panel of the JADE add-in, you will see an option labeled "List Automations." Click it to see a list of functions in the workbook that have been configured to be displayed in this list.



If you have followed all of the steps in this walk-through, you will see the following list of automations.

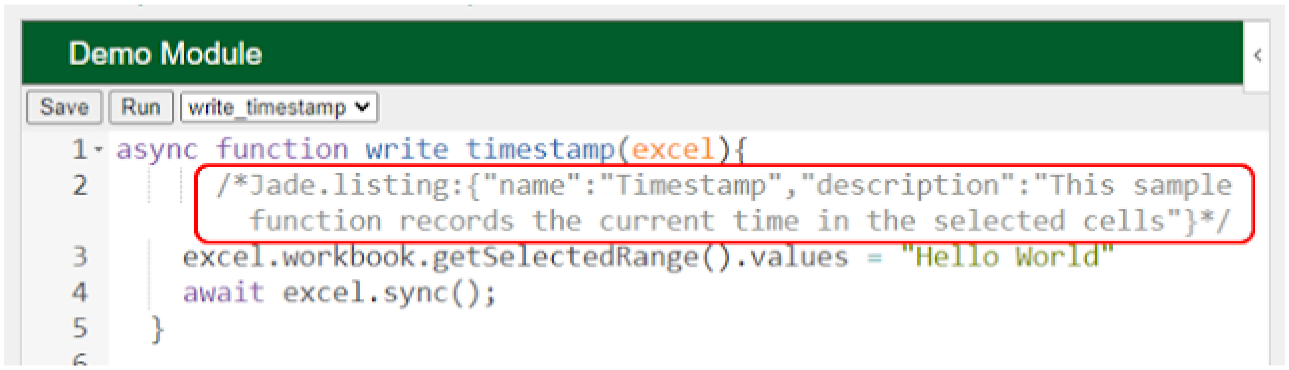


Active Automations

1. **Timestamp**: This sample function records the current time in the selected cells
2. **Create Worksheets**: Generate 10 sheets named for differnt kinds of apples
3. **Sort Worksheets**: Alphabetize the order of worksheets.
4. **Delete Worksheets**: Delete the 10 sheets named for differnt kinds of apples

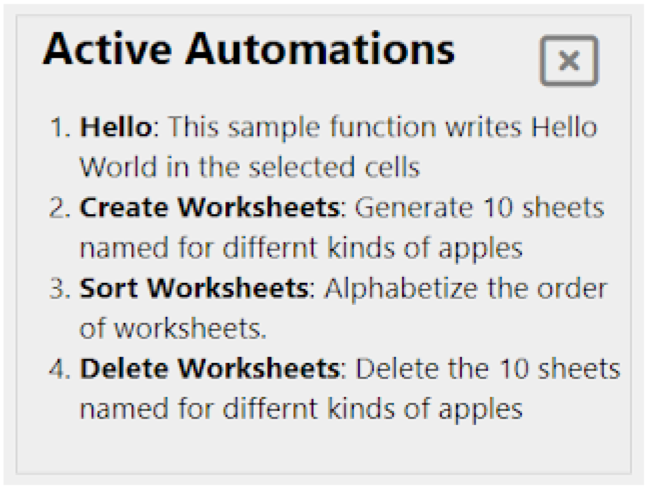
You can click the names of any of these automations to execute the code that they describe. Recall that we have edited the "write_timestamp" function to write "hello world" to the selected cell rather than the current date and time.

To update the listing for that function, close the Active Automations panel by clicking the "X" in the upper right corner to return to the "Home" panel. Then, click "Open Code Editor".



```
async function write_timestamp(excel){
    /*Jade.listing:{"name":"Timestamp","description":"This sample
      function records the current time in the selected cells"}*/
  excel.workbook.getSelectedRange().values = "Hello World"
  await excel.sync();
}
```

You will see that at the beginning of the function, there is a block comment that begins with "Jade.listing." In this comment, you can change "Timestamp" to "Hello" and the text that reads "records the current time" to "writes Hello World" to make the listing better reflect what the function actually does. Click "Save" to record the changes. The next time you choose "List Automations" from the Home panel, you will see the updated entry as follows.

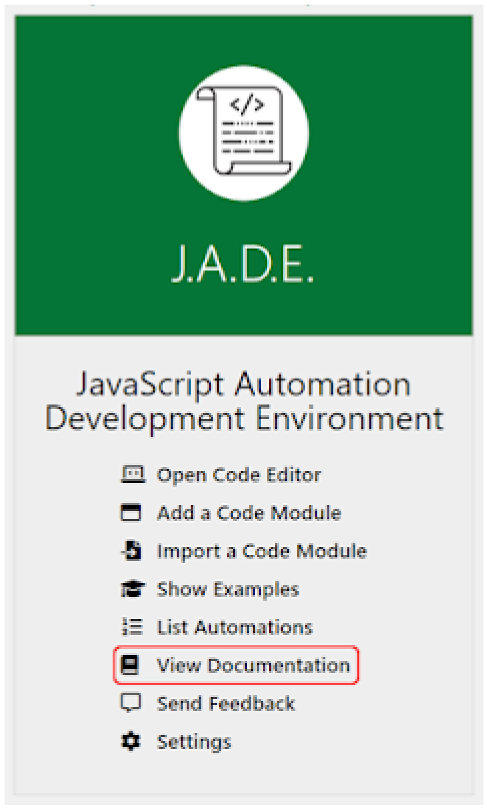


Active Automations

1. **Hello**: This sample function writes Hello World in the selected cells
2. **Create Worksheets**: Generate 10 sheets named for differnt kinds of apples
3. **Sort Worksheets**: Alphabetize the order of worksheets.
4. **Delete Worksheets**: Delete the 10 sheets named for differnt kinds of apples

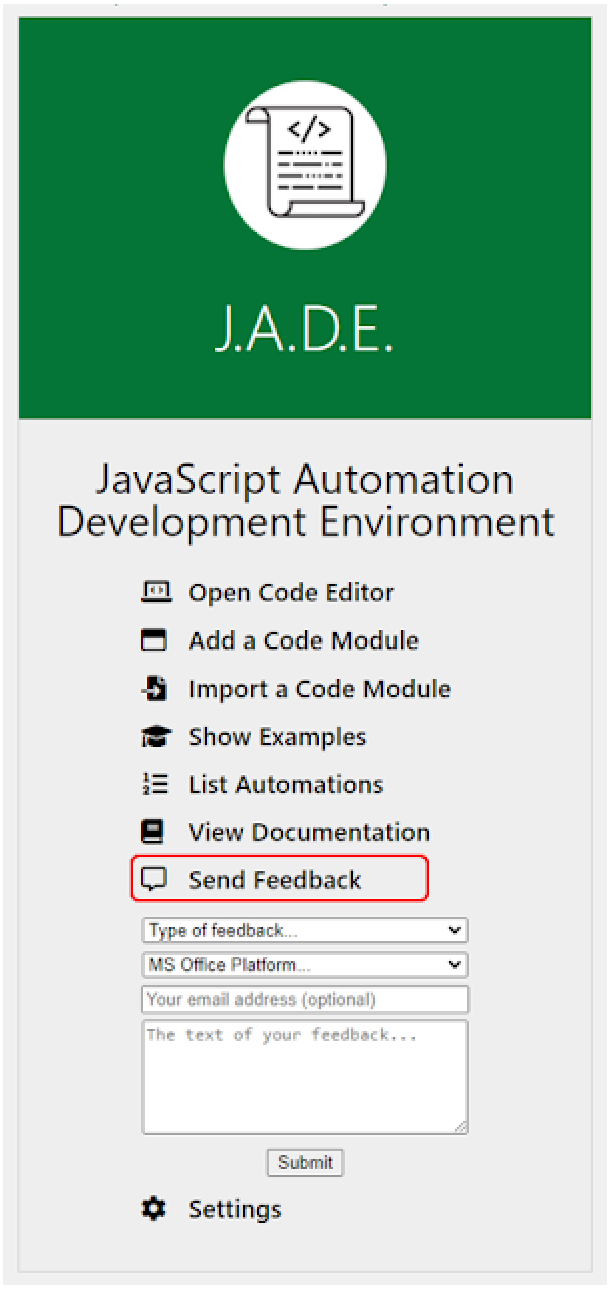
5. View Documentation

On the Home panel of the JADE add-in, you will see an option labeled "View Documentation." Click it to open the Documentation for the add-in (https://support.jsvba.com).



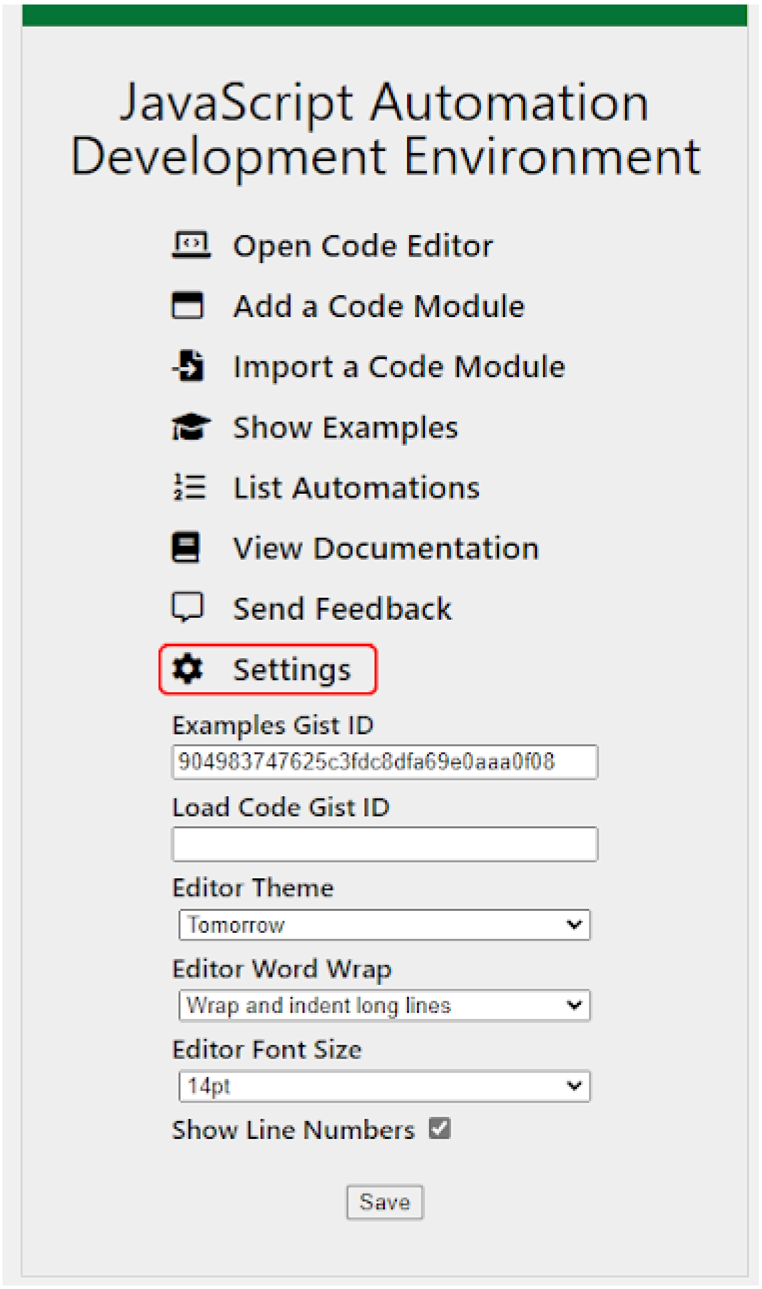
6. Send Feedback

On the Home panel of the JADE add-in, you will see an option labeled "Send Feedback." Click it to open a form to send feedback about the add-in to the development team as seen here.



7. Settings

On the Home panel of the JADE add-in, you will see an option labeled "Settings." Click it to reveal the settings you can change.


JavaScript Automation Development Environment

Examples Gist ID: 904983747625c3fdc8dfa69e0aaa0f08
Load Code Gist ID:
Editor Theme: Tomorrow
Editor Word Wrap: Wrap and indent long lines
Editor Font Size: 14pt
Show Line Numbers ✓

The settings available here are as follows:

1. **Examples Gist ID**: This is the ID of a Gist (or a comma-separated list of Gist IDs) that identifies the Gist that holds the examples. This is intended for use by instructors who may want to present a specific set of examples to students for a given class; However, the functionality is available to everyone. Just follow the patterns used in the default

examples and you can change the examples available in the add-in. You may have some code that you find yourself writing over and over. If so, you may want to build those patterns as a Gist and make them easily available to yourself during development.

2. **Load Code Gist ID**: This is the ID of a gist that has code to be loaded into memory when the add-in first opens. By specifying a Gist ID here, it is possible to do your development work on your local machine in your favorite coding environment. Once you publish your code modifications to the Gist, they will be available in the JADE add-in the next time the add-in is opened or refreshed. An example of this is shown in Section 9 below.

3. **Editor Theme**: This dropdown allows you to choose from several community-developed thems for the editor. Some have dark backgrounds while others have light ones. Play around with different options until you find one that meets your needs.

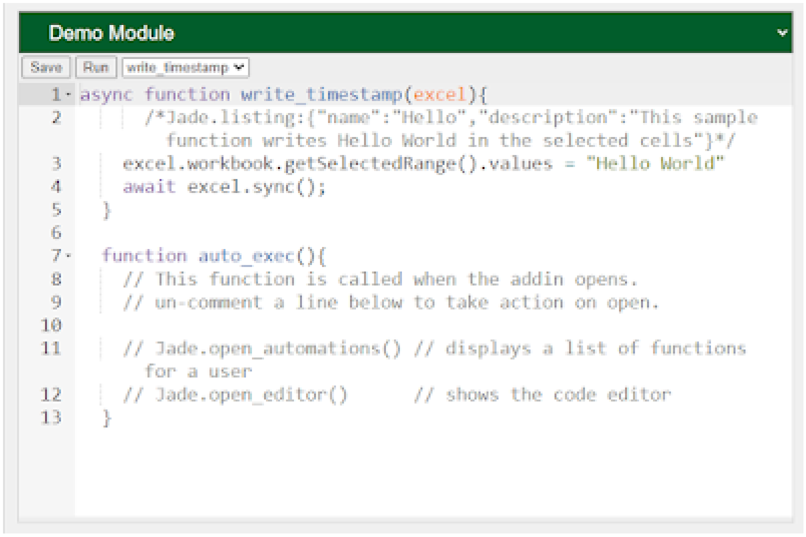
4. **Editor Word Wrap**: This provides three different options about how the editor should handle lines of code that are too long to be displayed fully with the current editor width,

5. **Editor Font Size**: This allows you to change the size of the font in the editor independently of the other text in the add-in.

6. **Show Line Numbers**: this option simply toggles whether line numbers are shown in the editor.

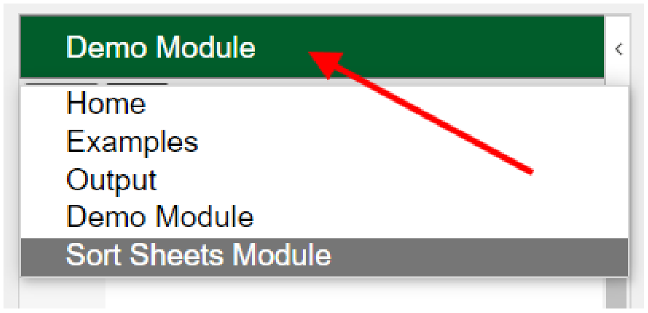
8. Output

There is one panel that is not directly available from the Home panel of the JADE add-in. It is called the "Output" panel. In a function, a user can call the "Jade.print" function to save information for later review. Examine the "Sort Sheets" module we imported in Section 2 (Import a Module) by clicking "Open Code Editor" on the Home panel. When you do, the editor will show the first module we created (In Section 1) as seen here.

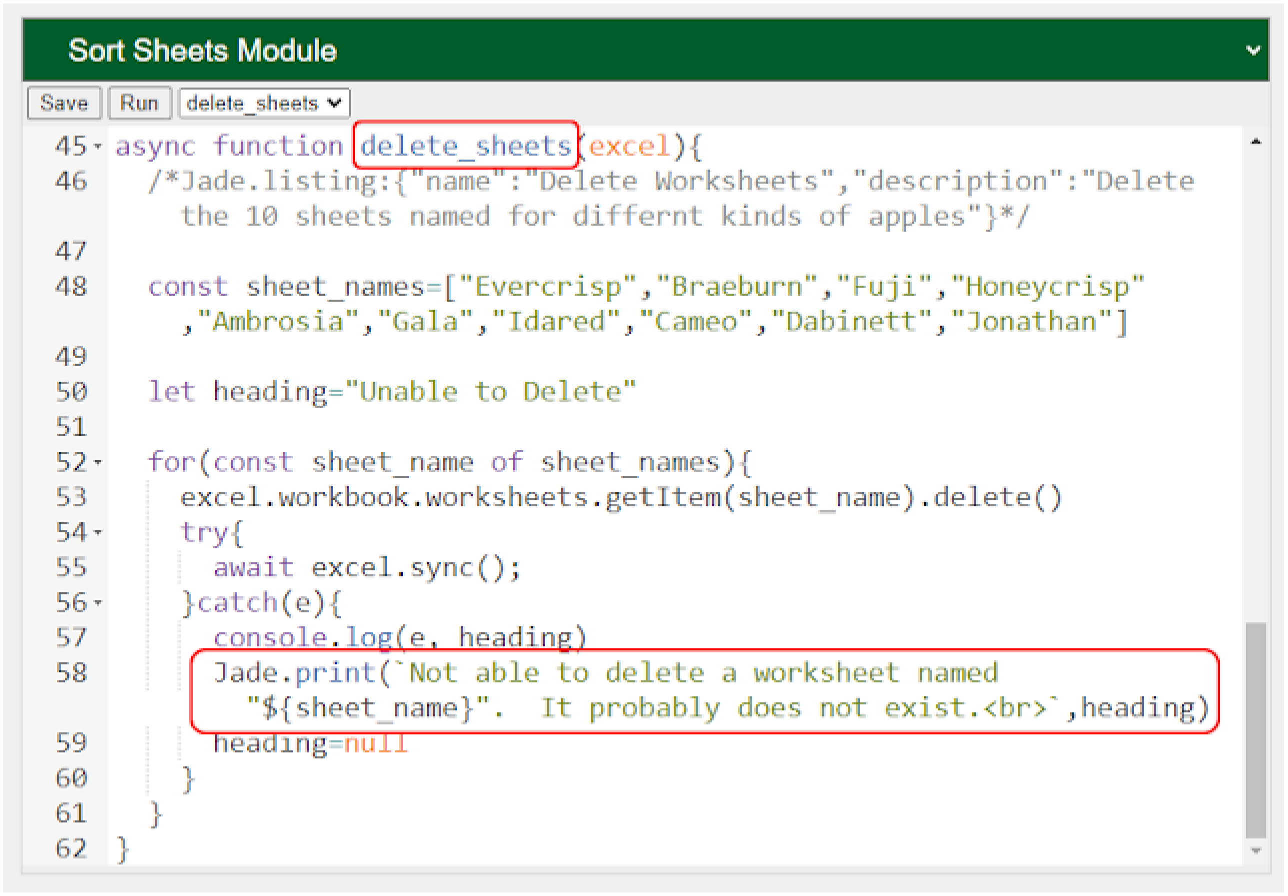


```
async function write_timestamp(excel){
    /*Jade.listing:{"name":"Hello","description":"This sample
      function writes Hello World in the selected cells"}*/
  excel.workbook.getSelectedRange().values = "Hello World"
  await excel.sync();
}

function auto_exec(){
  // This function is called when the addin opens.
  // un-comment a line below to take action on open.

  // Jade.open_automations() // displays a list of functions
    for a user
  // Jade.open_editor()      // shows the code editor
}
```

To get to the "Sort Sheets" module, click the green bar at the top of the add-in and choose "Sort Sheets Module"

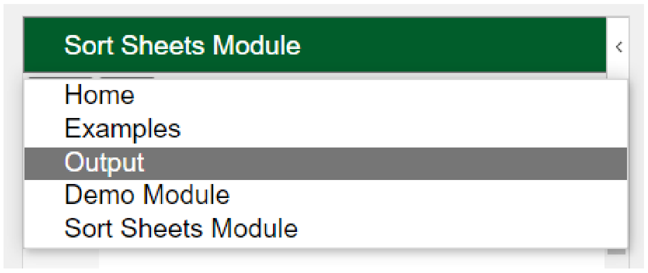


Now, scroll the editor window to the bottom of the module so you can see the function named "delete_sheets"

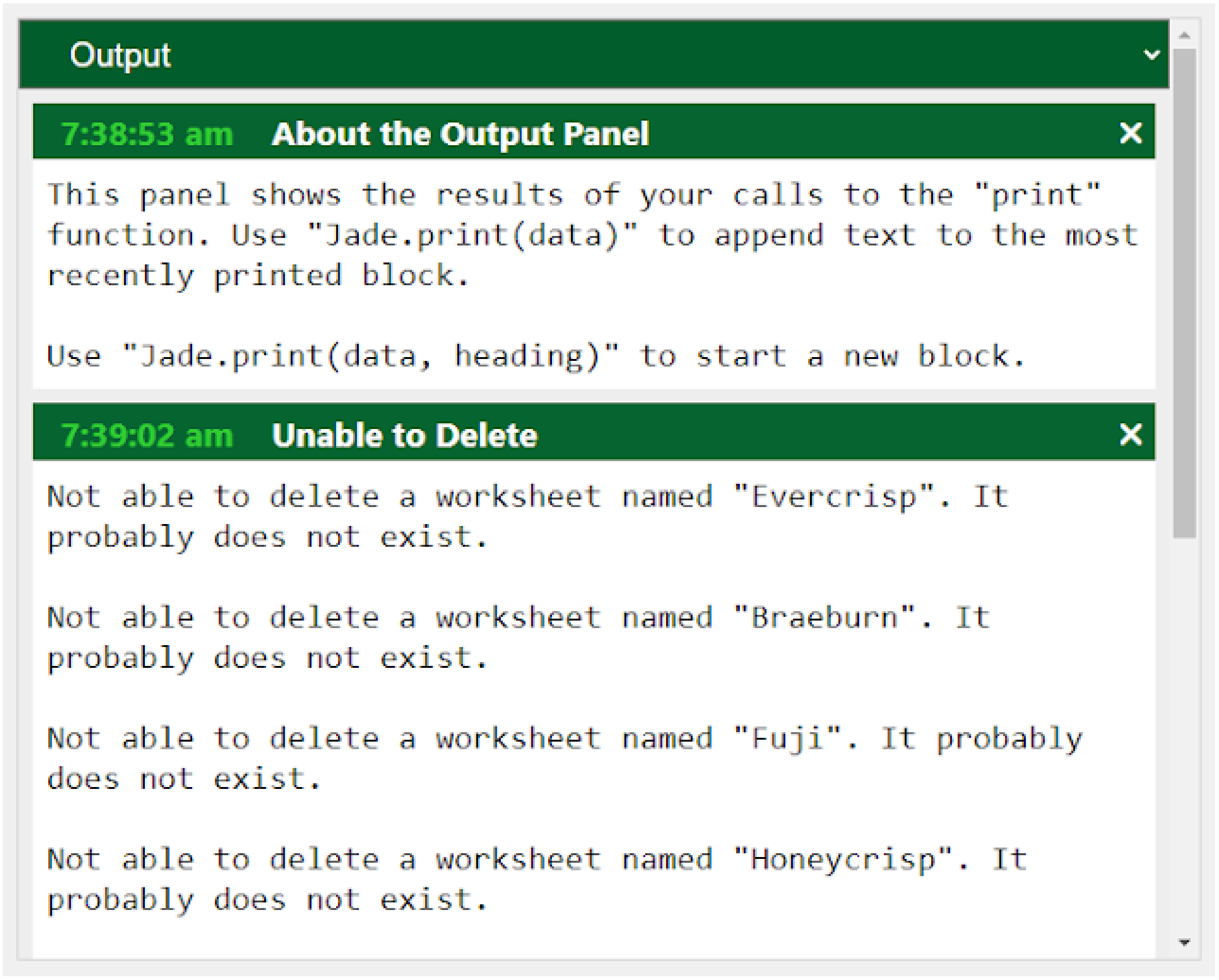


```
async function delete_sheets(excel){
  /*Jade.listing:{"name":"Delete Worksheets","description":"Delete
    the 10 sheets named for differnt kinds of apples"}*/

  const sheet_names=["Evercrisp","Braeburn","Fuji","Honeycrisp"
    ,"Ambrosia","Gala","Idared","Cameo","Dabinett","Jonathan"]

  let heading="Unable to Delete"

  for(const sheet_name of sheet_names){
    excel.workbook.worksheets.getItem(sheet_name).delete()
    try{
      await excel.sync();
    }catch(e){
      console.log(e, heading)
      Jade.print(`Not able to delete a worksheet named
        "${sheet_name}".  It probably does not exist.<br>`,heading)
      heading=null
    }
  }
}
```

This function tries to delete worksheets with the names that were created by the "create_sheets" function. However, it is configured to notice when the attempt to delete the sheet fails. It uses "Jade.print" to log that information to the Output panel. To observe this, just run the "delete_sheets" function twice. Then show the Output panel by clicking the green bar at the top of the code editor and selecting "Output"



You will see output similar to the following:



7:38:53 am About the Output Panel

```
This panel shows the results of your calls to the "print"
function. Use "Jade.print(data)" to append text to the most
recently printed block.

Use "Jade.print(data, heading)" to start a new block.
```

7:39:02 am Unable to Delete

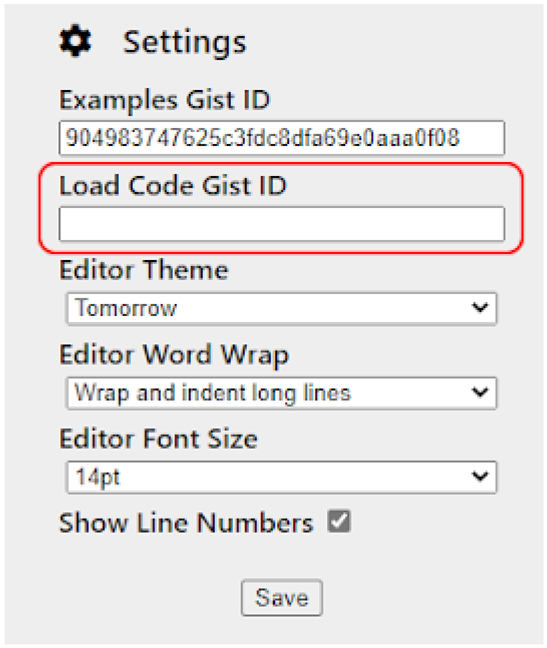
```
Not able to delete a worksheet named "Evercrisp". It
probably does not exist.

Not able to delete a worksheet named "Braeburn". It
probably does not exist.

Not able to delete a worksheet named "Fuji". It probably
does not exist.

Not able to delete a worksheet named "Honeycrisp". It
probably does not exist.
```

9. Load Code from Gist

The simplest way to save JADE automation code is directly in the workbook. This works well for individual users developing automations for personal use. However, if the automation code is to be used by many people, it becomes difficult to update the code that is stored in many different workbooks around an organization. Another option is to store the code in a central repository so updates only need to be made in one location. By storing the code for an automation or set of automations in a gist on github.com, either public or secret, centralized code management can be achieved.

To follow this example, begin by making a new Excel workbook and opening the JADE add-in.

To consume code from a Gist, click "Settings" on the JADE add-in Home panel.


Settings

Examples Gist ID: 904983747625c3fdc8dfa69e0aaa0f08

Load Code Gist ID:

Editor Theme: Tomorrow

Editor Word Wrap: Wrap and indent long lines

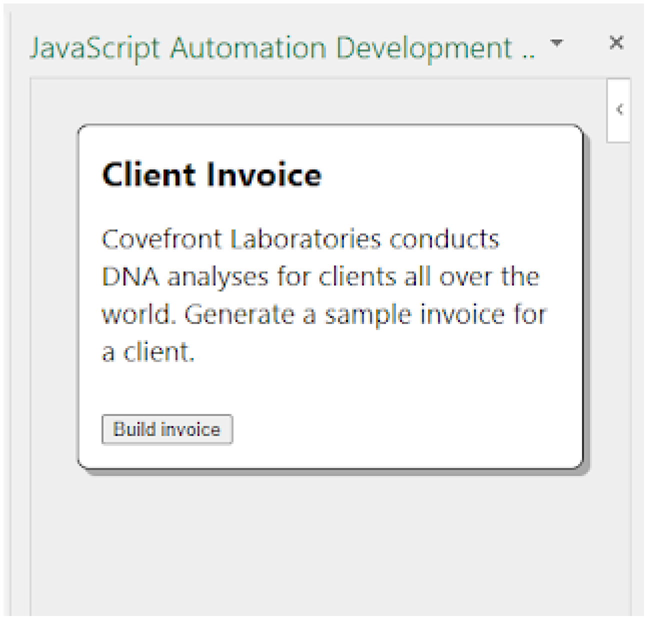
Editor Font Size: 14pt

Show Line Numbers ☑

Now, enter the following Gist ID in the "Load Code Gist ID" field and click "Save".

89042ed903c6f9368d23f0b55976540c

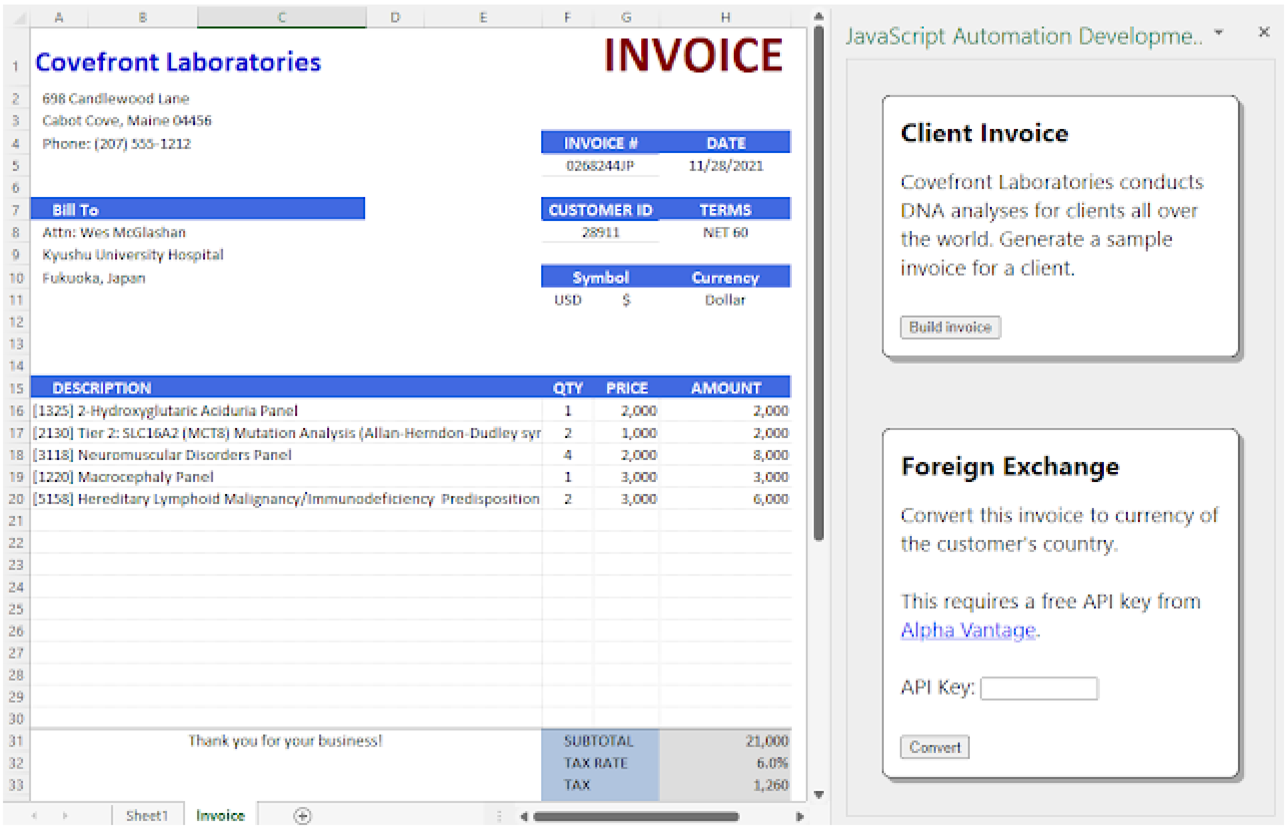
Because this module has an "auto_exec" function that initializes the automation, the next time the JADE add-in is opened on this workbook, the user interface for this automation will be displayed, instead of the JADE Home panel. To see this, just close and re-open the JADE add-in. You will see the following.



Client Invoice

Covefront Laboratories conducts DNA analyses for clients all over the world. Generate a sample invoice for a client.

This example builds an invoice for Covefront Laboratories, a fictional company that conducts DNA testing for clients all over the world. Click the "Build Invoice" button to generate a sample invoice, which will appear similar to the following:



Covefront Laboratories **INVOICE**

698 Candlewood Lane
Cabot Cove, Maine 04456
Phone: (207) 555-1212

INVOICE #	DATE
0268244JP	11/28/2021

Bill To
Attn: Wes McGlashan
Kyushu University Hospital
Fukuoka, Japan

CUSTOMER ID	TERMS
28911	NET 60

Symbol		Currency
USD	$	Dollar

DESCRIPTION	QTY	PRICE	AMOUNT
[1325] 2-Hydroxyglutaric Aciduria Panel	1	2,000	2,000
[2130] Tier 2: SLC16A2 (MCT8) Mutation Analysis (Allan-Herndon-Dudley syr	2	1,000	2,000
[3118] Neuromuscular Disorders Panel	4	2,000	8,000
[1220] Macrocephaly Panel	1	3,000	3,000
[5158] Hereditary Lymphoid Malignancy/Immunodeficiency Predisposition	2	3,000	6,000

Thank you for your business!	SUBTOTAL	21,000
	TAX RATE	6.0%
	TAX	1,260

Client Invoice

Covefront Laboratories conducts DNA analyses for clients all over the world. Generate a sample invoice for a client.

Foreign Exchange

Convert this invoice to currency of the customer's country.

This requires a free API key from Alpha Vantage.

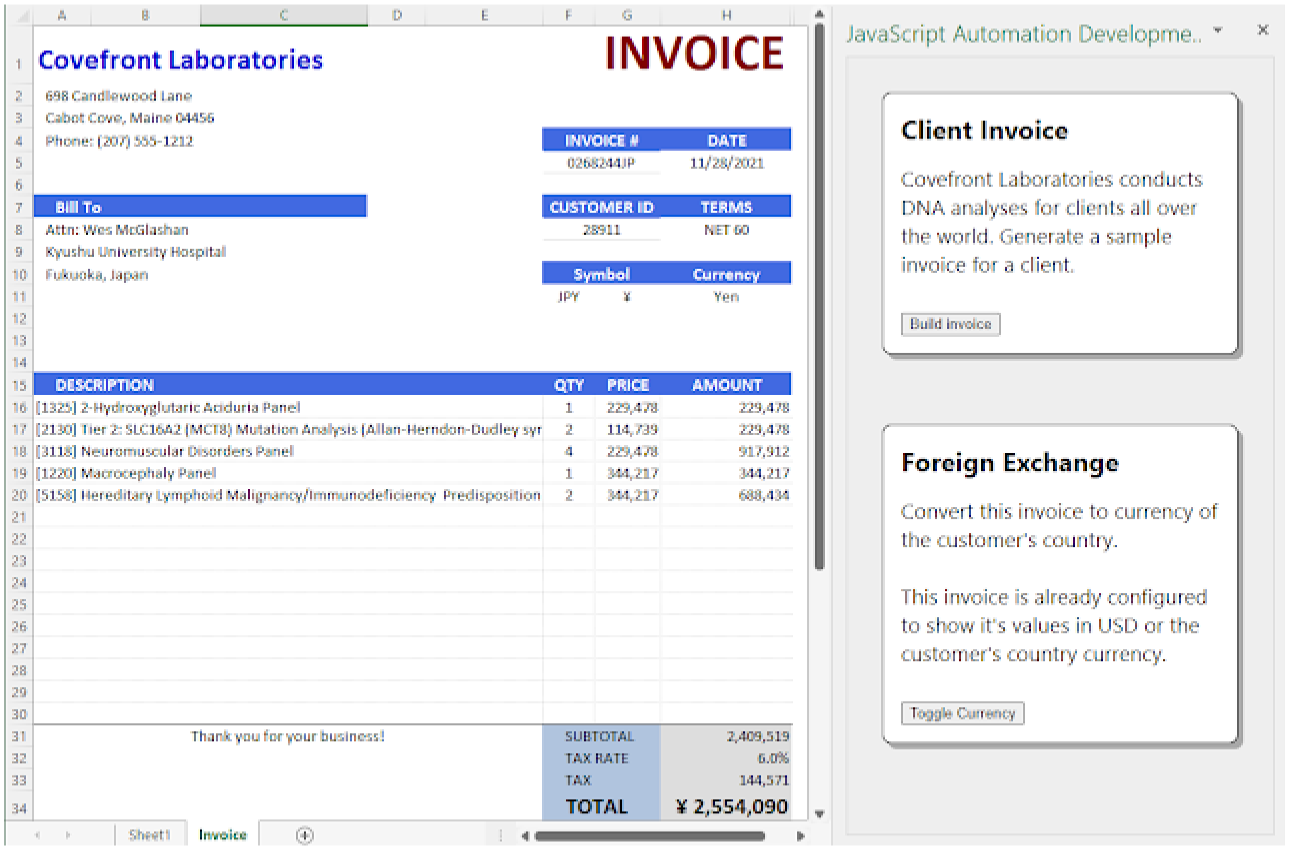
API Key:

Because this customer is from a country other than the United States, the "Foreign Exchange" pane is displayed by the automation. If the invoice you generated is for a customer in the United States, the Foreign Exchange pane will not appear. Simply click "Build Invoice" again until you get one from a different country.

This automation can convert the invoice to the currency of the country of the customer using the current exchange rate retrieved by making a call to a foreign exchange API provided by Alpha Vantage. To convert the invoice, enter the following code in the API Key field and click the "Convert" button.

G0O6KVPUQ0M3960Y

Now your invoice should be converted to the client's native currency as seen here.



Covefront Laboratories

INVOICE

698 Candlewood Lane
Cabot Cove, Maine 04456
Phone: (207) 555-1212

INVOICE #	DATE
0268244JP	11/28/2021

Bill To
Attn: Wes McGlashan
Kyushu University Hospital
Fukuoka, Japan

CUSTOMER ID	TERMS
28911	NET 60

Symbol		Currency
JPY	¥	Yen

DESCRIPTION	QTY	PRICE	AMOUNT
[1325] 2-Hydroxyglutaric Aciduria Panel	1	229,478	229,478
[2130] Tier 2: SLC16A2 (MCT8) Mutation Analysis (Allan-Herndon-Dudley syr	2	114,739	229,478
[3118] Neuromuscular Disorders Panel	4	229,478	917,912
[1220] Macrocephaly Panel	1	344,217	344,217
[5158] Hereditary Lymphoid Malignancy/Immunodeficiency Predisposition	2	344,217	688,434

Thank you for your business!

SUBTOTAL	2,409,519
TAX RATE	6.0%
TAX	144,571
TOTAL	**¥ 2,554,090**

Client Invoice

Covefront Laboratories conducts DNA analyses for clients all over the world. Generate a sample invoice for a client.

Foreign Exchange

Convert this invoice to currency of the customer's country.

This invoice is already configured to show it's values in USD or the customer's country currency.

Now that the exchange rate has been written to the worksheet, you can toggle the currency between US dollars and the currency of the client's country by clicking the "Toggle Currency" button.

The important thing to realize here is that the code for this example is hosted by github.com and is read into this workbook each time the add-in is opened or refreshed. The code is not imported into the workbook, it's just loaded into memory in the add-in.

10. Conclusion

Well, that's it. Should you want to contact the developer, you can reach out to Gove Allen by phone (801-372-0683) or by email (gove@colonialheritage.org)