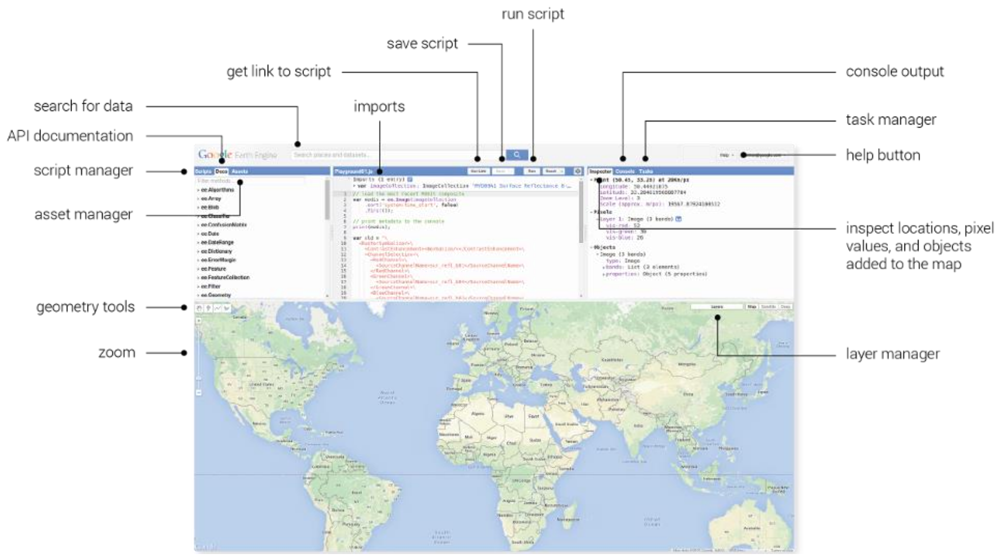
Exercise 1: Introduction to the Code Editor and Data Archive



Introduction

Google Earth Engine is a cloud-based geospatial processing platform. It is available through Python and JavaScript Application Program Interfaces (APIs). The JavaScript API is accessible via a web-based Integrated Development Environment (IDE) called the Code Editor which is what you'll be using for this training. The Code Editor offers access to the full power of Earth Engine. This platform is where users can write and execute scripts to share and repeat geospatial analysis and processing workflows.

In this exercise, you will learn about the Code Editor platform and explore some basic scripting concepts in JavaScript. Some basic coding and JavaScript knowledge is required to use the Earth Engine Code Editor. If you've never written any code before and this tutorial is too advanced, you may want to look at GTAC's *Introduction to Geospatial Scripting* course. This course has some reminder and overview of basic scripting, but that material is not covered in depth.

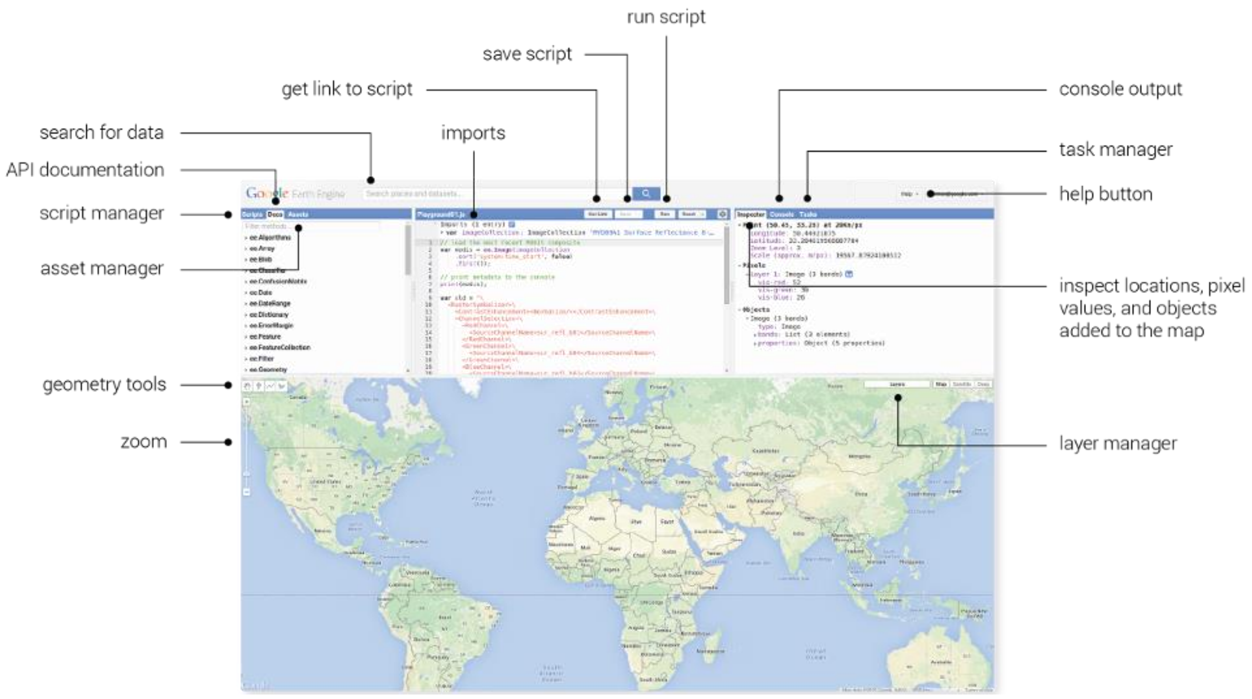
Table of Contents

Part 1: Introduction to the Code Editor IDE

In this exercise, you will work in the Google Earth Engine Code Editor. This platform offers significantly more flexibility than the GUI based Explorer platform. You have the ability to create complex and customized analysis workflows. In the Code Editor you will write JavaScript code to access and analyze imagery. Because of Earth Engine's cloud computing architecture, there are some unique JavaScript classes that will also be covered in this course.

A. Explore the JavaScript Code Editor

1. In your Google Chrome web browser navigate to the following URL: https://code.earthengine.google.com/
   i. When/if prompted, click **Allow** to let the Earth Engine Code Editor access your Google Account.
   ii. This will take you to the Code Editor interface shown below.



2. Use the graphic above to guide you and click through the tabs in the upper left hand Scripts and Documentation panel.
   i. Under the Scripts tab in the upper left, click the **Examples** dropdown and note the wide variety of preloaded example scripts that demonstrate capabilities and offer code that you can use for your analyses. You can take a look at these to start to learn about what kinds of things Earth Engine can do. After you create and save a script later in the training, it will be available here in your Private repository.
   ii. Click the **Docs** tab, also in the upper left, there is a searchable list of documentation for the predefined GEE classes and methods. Note these are grouped and organized by class.

iii. Briefly explore what kinds of methods and functions are available in GEE for different classes.

(a) Select one of interest and **click** on it to see the information window with a description of the methods and associated arguments (required and optional). Any optional arguments are italicized. (The example scripts include examples of many of Earth Engine methods and their arguments. Try **searching** for them using the scripts search bar.)

3. Using the graphic above **click through** the tabs in the upper right hand panel where the Inspector, Console, and Tasks tabs are located.

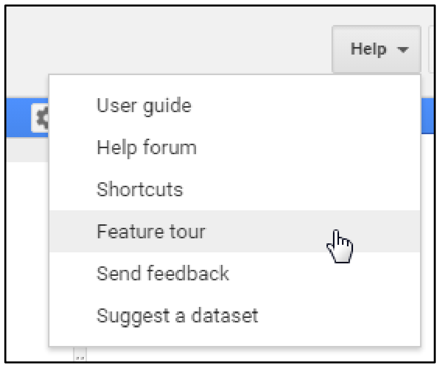
   i. You will use the Inspector (similar to the identify tool in ArcMap) to easily get information about layers in the map at specified points (specified by clicking in the Map Panel).

   ii. The Console is used to return messages as the scripts run and print information about the data, intermediate products and results. It also records any diagnostic messages, such as information about runtime errors.

   iii. The Tasks tab is used to manage the exporting of data and results.

4. Click on the **Help** button in the upper right and select Feature Tour to learn more about each component of the API.

   i. **Click through** the options in the Feature tour to become more familiar with each component of the Code Editor.

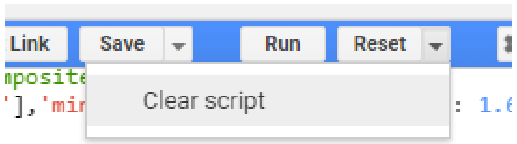


Part 2: Working with Images

A. Open a new script

1. Open the Code Editor webpage in Google Chrome, if it is not already open: https://code.earthengine.google.com/

   i. If you already have the code editor open, but have a script loaded, click on the dropdown arrow adjacent to the Reset button and select **Clear script**.



B. Create a variable representing a single Landsat 8 image

1. Use the code in the box below to create a variable representing an ee.Image object for a 2014 Landsat 8 image.
   i. **Copy and paste** the code below into the Code Editor Code Editor.

```
// Get the image.
var lc8_image = ee.Image('LANDSAT/LC8_L1T_TOA/LC80450322014244LGN00');
```

Notes about JavaScript syntax: *There's a lot going on in just two lines. Take a closer look at the pieces of this statement you're loading into GEE (remember that if this statement is completely foreign to you, you should consider looking at the Introduction to Geospatial Scripting course for a more detailed explanation of the scripting language). The numbers below give you a quick reminder of some of the important points in the two lines.*

*1) Double forward slashes, **//**, are comment characters in JavaScript. These prevent text on that line from executing. These are useful for creating notes in your code.*

*2) Variables are declared in JavaScript using the keyword **var**. Variables can be numbers, strings, images, features, etc. Variables are used to store information for use later on in the script. In the case of the statement above, you are naming the variable **lc8_image** and using it to refer to the raster dataset you are interested in analyzing.*

*3) **ee.Image()** is a GEE class designation that tells GEE that you want to load an ee.Image object (and in this case, save it as a variable called 'lc8_image'). Earth Engine classes all begin with ee. The parentheses at the end let you define which object of the ee.Image class type you're loading from the data catalog. In this case, the parameter you are specifying inside the parentheses is the image ID.*

*A generalized version of the statement above is: **ee.Image('image_id')**. The 'image_id' is the image that you would like to load ('LANDSAT/LC8_L1T_TOA/LC81290502015036LGN00') and reference with the variable (lc8_image).*

*4) The syntax for specifying the image ID in this function (**ee.Image**) is to surround the string of characters (the image ID, 'LANDSAT/LC8_L1T_TOA/LC81290502015036LGN00') in quotes. The image id is in quotes because the collection and image name together is a **string**. **Strings** are sets of characters that in this example, name the specific dataset.*

5) JavaScript statements end with a semicolon.

Click the **Run** button and note that nothing happens in the map or the console. This code merely creates the variable, nothing is printed or displayed.

C. Add the image to the Code Editor map

1. Copy and paste, or type, the code below into your script. These additional lines will add the Landsat image to the map panel. Add these lines *below* the code from the previous step. GEE will execute the code (lines) sequentially when you hit Run.

```
// Add the image to the map.
Map.addLayer(lc8_image);
```

2. Click on the **Run** button. This time an image will load in the Map Output window. If you are not zoomed into the USA, centered on California, you won't see anything.
   i. Use your cursor to **navigate** (left click and drag) in the map view to California and find the image you called. It would be nice if the script did this for us, next you will add that statement into your script.

D. Center and Zoom the map window

Next you will add a statement to set the zoom factor and location on which to center the map output window. Map.centerObject() is a function that tells GEE where to position the map output window.

1. **Copy and paste** the two lines of code (below) underneath the four lines you already have in the GEE code editor window.
2. Click **Run**.

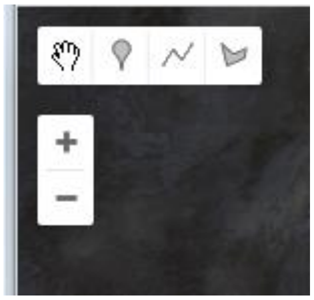
```
// Center the map display on the image.
Map.centerObject(lc8_image, 8);
```

3. To zoom out, decrease the second input to a number less than 8. To zoom in more, increase the second input parameter (try 10). **Modify** your statement so it looks like the two lines below and
4. Click **Run**. What happened?
5. In the panel in the upper left click the **Docs** tab. Type **Map.centerObject()** into the Docs search bar. What is the range of the zoom parameter?

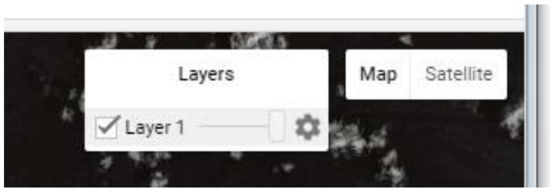
```
// Center map display on the image.
Map.centerObject(lc8_image, 10);
```

E. Explore the map window tools

1. Explore this image with the Code Editor map viewer tools.
   i. You can zoom and pan using the tools on the left of the map output panel (shown in the following graphic).



   ii. Click the **check box** Layers tool (on the right of the map output panel) to turn the image (Layer 1) on or off (shown in following image).



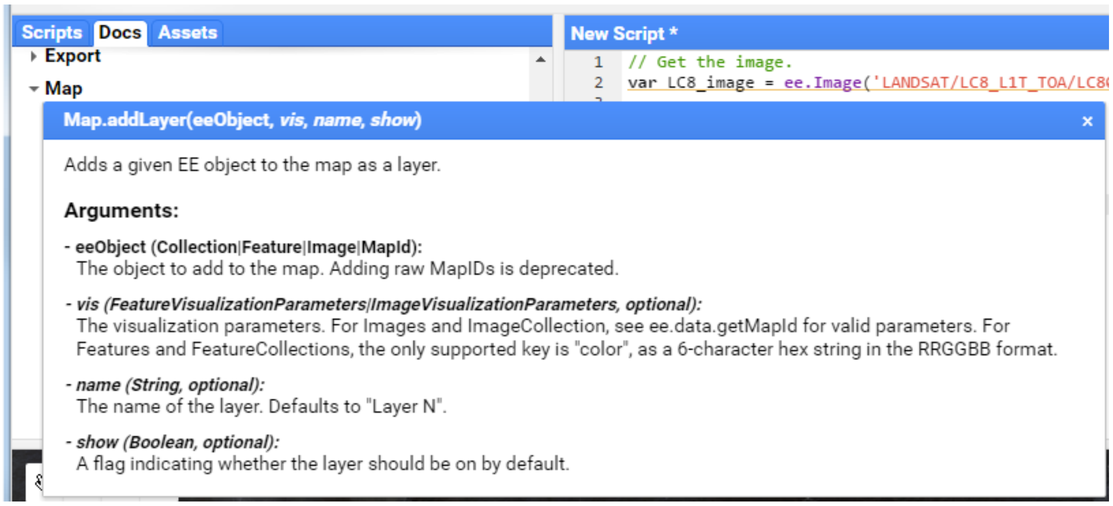
***Note:** Even though you saved the **LANDSAT/LC8_L1T_TOA/LC80450322014244LGN00** image as a variable named **lc8_image**, the name of the image is labeled as **Layer_1** by default in the Layers Legend in the output map window. As you will see later, it is possible to change the name that appears in the Layers tool to something that is more descriptive of the data being displayed.*

iii. Swipe the transparency lever (the sliding bar to the right of the layer name in the preceding image). This will make the 'Layer 1' transparent, revealing the base map underneath.

F. Change visualization parameters to improve the display

Now you can see the image, however the color parameters are not very well suited to this image. You will change the visualization parameters next.

1. Open the documentation for addLayer function in the Map group by clicking on the **Docs** tab in the left panel in the Code Editor.
   i. **Expand** the Map group and select Map.addLayer from the list (as illustrated below); or
   ii. Search for Map.addLayer in the Filter methods… search bar.



```
// Get the image.
var LC8_image = ee.Image('LANDSAT/LC8_L1T_TOA/LC8
```

Map.addLayer(eeObject, *vis, name, show*)

Adds a given EE object to the map as a layer.

Arguments:

- **eeObject (Collection|Feature|Image|MapId):**
  The object to add to the map. Adding raw MapIDs is deprecated.
- ***vis (FeatureVisualizationParameters|ImageVisualizationParameters, optional):***
  The visualization parameters. For Images and ImageCollection, see ee.data.getMapId for valid parameters. For Features and FeatureCollections, the only supported key is "color", as a 6-character hex string in the RRGGBB format.
- ***name (String, optional):***
  The name of the layer. Defaults to "Layer N".
- ***show (Boolean, optional):***
  A flag indicating whether the layer should be on by default.

2. Review the documentation that appears (shown above). This provides information about the use and arguments for this function.
   i. Notice that some input options (such as *vis*) are italicized in the documentation. This means that these are optional parameters that can be specified or left out of the Map.addLayer statement. If you want to skip an optional parameter use "undefined" as a place holder. See the statement below for an example.

```
// Add the image to the map and name the layer in the map window.
Map.addLayer(lc8_image, undefined, 'Landsat8scene');
```

There are a number of options available to adjust how images are displayed. The inputs that are most commonly used to modify display settings include the following:

***Bands:** allows the user to specify which bands to render as red, green, and blue.*
***Min and max:** sets the stretch range of the colors. The range is dependent on the data type. For example unsigned 16-bit imagery has a total range of 0 to 65,536. This option lets you set the display to a subset of that range.*
***Palette:** specifies the color palette used to display information. You will see how to use this option later in the tutorial.*
***Naming convention (in the Layers Legend):** you can specify the name that appears in the layers legend here as well. You named this layer **'Landsat8scene'** in the code above.*

***Syntax:** Most of these optional parameters are entered as a key-value pair in a dictionary object. The syntax is:*

{vis_param1: number, number, number

vis_param2: 'string, string, string',

// or an array of strings like this:

vis_param2: ['string', 'string', 'string']}

3. **Modify** the Map.addLayer() function to display the image as a false color composite and apply a stretch to improve the display. Modify the Map.addLayer() statement from the previous steps to look like the code below. The statement below includes the optional parameters for which bands to display (bands 6, 5, and 4), specifies a stretch to improve the visualization, and finally gives the image a display name.

```
// Add the image to map as a false color composite.
Map.addLayer(lc8_image, {bands: 'B6,B5,B4', min: 0.05, max: 0.8,
gamma: 1.6}, 'Landsat8falseColor');
```

4. Click **Run** and use the map tools to explore the result. Note that the name under the Layers (legend) is now Landsat8falseColor.

Note: In the statement above, the names of the bands have been inserted for you already. If you wanted to look them up yourself, you can use the print function (or the Inspector) to identify what the bands are named (e.g., B6, B5, B4).

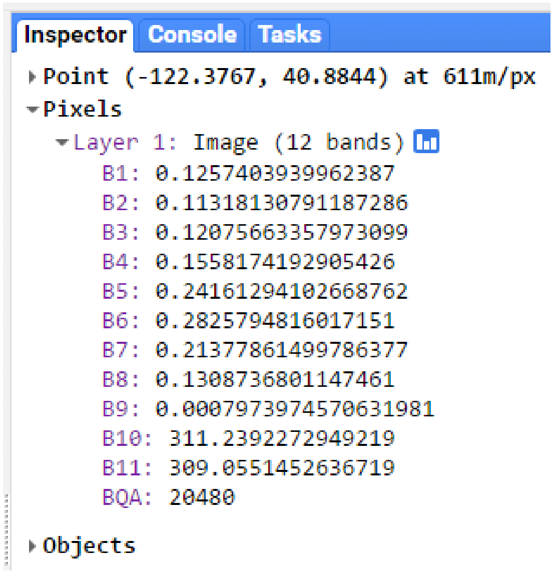
5. You can also specify the bands as strings in a list. Look at the statement below, it will do the same thing as the statement above. Do you notice the difference in syntax?

```
// Add the image to map as a false color composite.
```

```
Map.addLayer(lc8_image, {bands: ['B6', 'B5', 'B4'], min: 0.05, max:
0.8, gamma: 1.6}, 'Landsat8falseColor');
```

G. Explore the Inspector window

1. Click on the **Inspector Tab** in the upper right hand corner of the Earth Engine Code Editor interface. Your cursor will now change to a cross hairs when you place it in the map window.
2. Now **click anywhere on the map** using the Inspector (the cross hairs) to identify pixel values for each band in your image at the selected location. Refer to following graphic for example output.



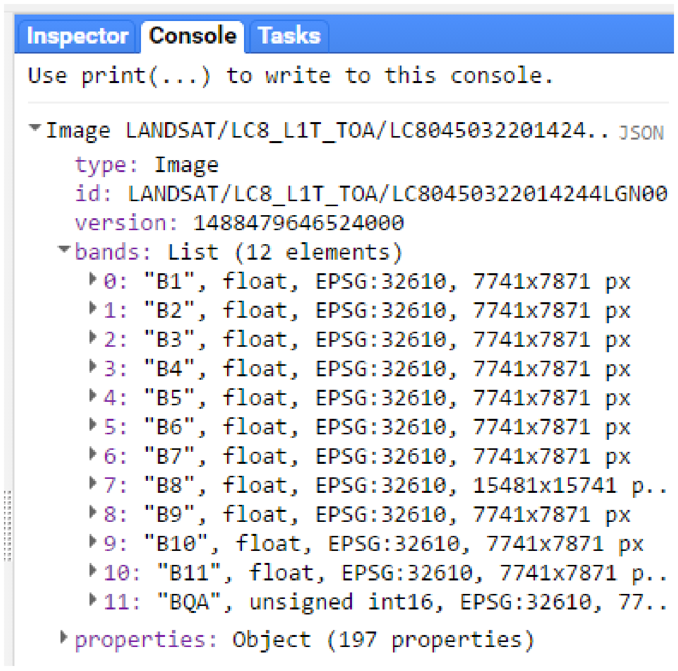
```
▸Point (-122.3767, 40.8844) at 611m/px
▾Pixels
  ▾Layer 1: Image (12 bands)
      B1: 0.1257403939962387
      B2: 0.11318130791187286
      B3: 0.12075663357973099
      B4: 0.1558174192905426
      B5: 0.24161294102668762
      B6: 0.2825794816017151
      B7: 0.21377861499786377
      B8: 0.1308736801147461
      B9: 0.0007973974570631981
      B10: 311.2392272949219
      B11: 309.0551452636719
      BQA: 20480

▸Objects
```

3. Now examine information about the image in the console. You'll need to use a print statement. Copy and paste the following statement in your code editor. Then click Run.

```
// Print the image information.
print(lc8_image);
```

4. Now in the Console tab, click on the **arrow** next to Image LANDSAT/… to display the image properties. Then click on the arrow next to bands: to display the band properties. This will reveal that the first band (indexed at 0) is called “B1”, the second (indexed at 1) is called “B2”, etc. Refer to following graphic for an example.



```
Inspector  Console  Tasks
Use print(...) to write to this console.

▾Image LANDSAT/LC8_L1T_TOA/LC8045032201424.. JSON
   type: Image
   id: LANDSAT/LC8_L1T_TOA/LC80450322014244LGN00
   version: 1488479646524000
  ▾bands: List (12 elements)
    ▸0: "B1", float, EPSG:32610, 7741x7871 px
    ▸1: "B2", float, EPSG:32610, 7741x7871 px
    ▸2: "B3", float, EPSG:32610, 7741x7871 px
    ▸3: "B4", float, EPSG:32610, 7741x7871 px
    ▸4: "B5", float, EPSG:32610, 7741x7871 px
    ▸5: "B6", float, EPSG:32610, 7741x7871 px
    ▸6: "B7", float, EPSG:32610, 7741x7871 px
    ▸7: "B8", float, EPSG:32610, 15481x15741 p..
    ▸8: "B9", float, EPSG:32610, 7741x7871 px
    ▸9: "B10", float, EPSG:32610, 7741x7871 px
    ▸10: "B11", float, EPSG:32610, 7741x7871 p..
    ▸11: "BQA", unsigned int16, EPSG:32610, 77..
  ▸properties: Object (197 properties)
```

Note: *To learn about band combinations for viewing Landsat 8 and Landsat 5 or 7, check out this comparison of Landsat bands: http://landsat.usgs.gov/L8_band_combos.php.*

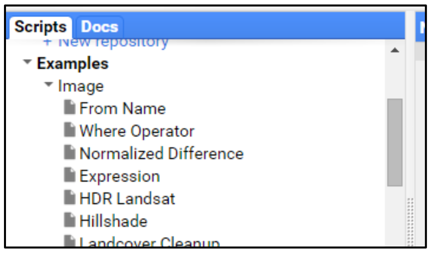
5. Click the **Save** button in the upper right of the Code Editor panel to save your example script for future reference.
   i. Name this script "Visualize a Landsat 8 image".

Note: *There are many more options for visualizing data in the map window, such as setting a mask or mosaicking two data sets together.*

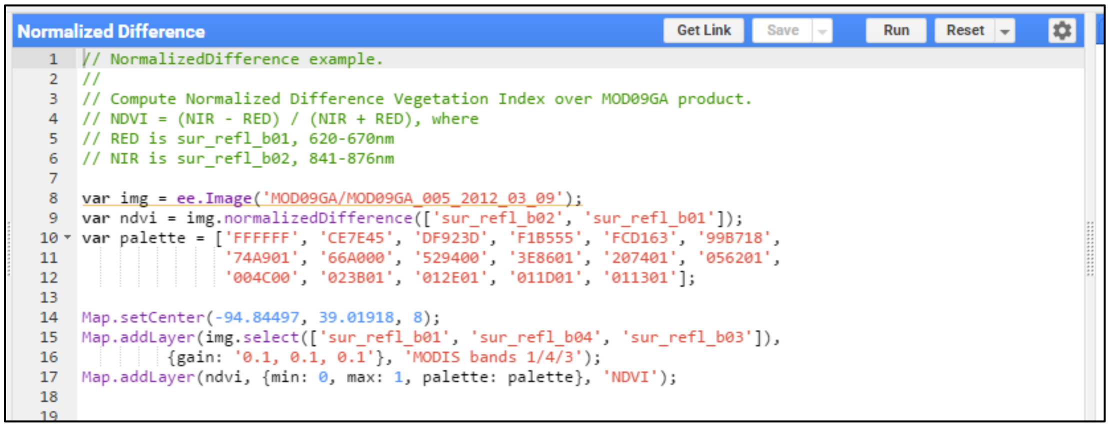
Part 3: Run an example script and review the results

A. The Examples Section

1. Click on the **Scripts tab** in the left-hand panel and expand the Examples group.
2. Scroll down until you see the Image group. Click on the triangle to expand this group if necessary.



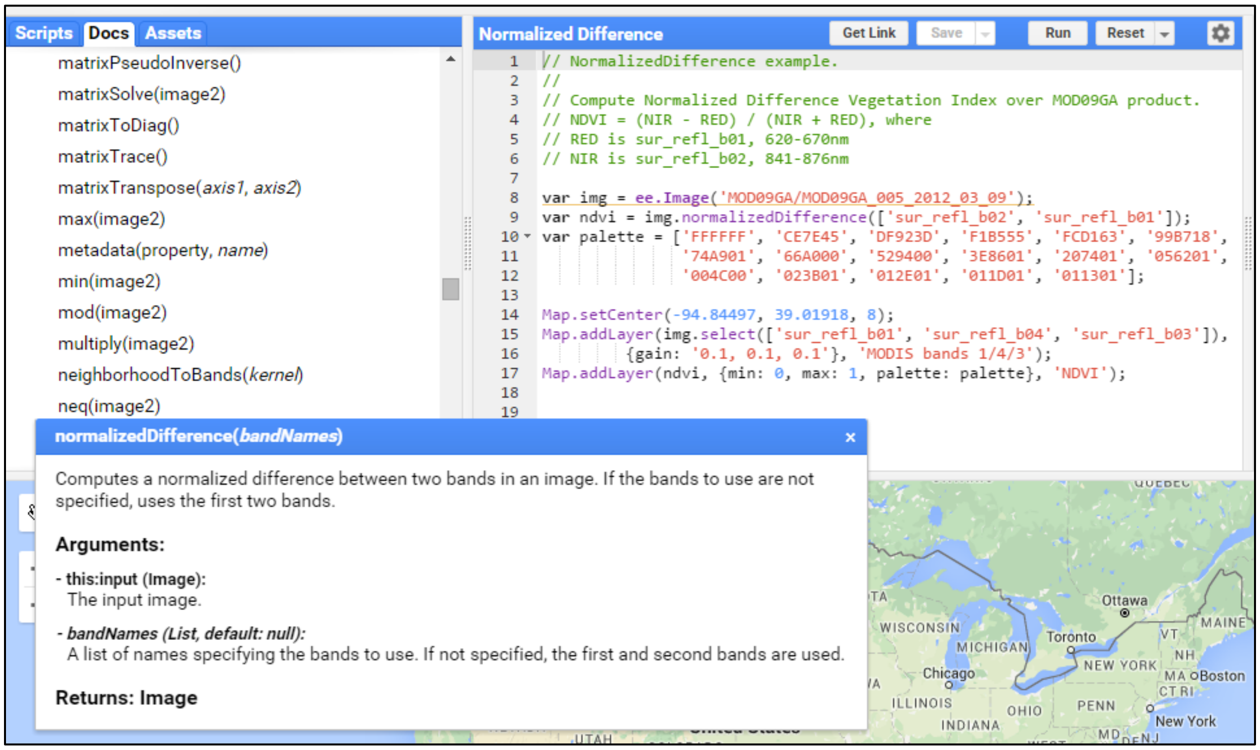
3. Select the **Normalized Difference** script from the list of example scripts (stored within the Image group). It will copy the script into your Code Editor panel.
4. The graphic below shows the script that should appear in the Code Editor panel (upper center panel).



```
 1  // NormalizedDifference example.
 2  //
 3  // Compute Normalized Difference Vegetation Index over MOD09GA product.
 4  // NDVI = (NIR - RED) / (NIR + RED), where
 5  // RED is sur_refl_b01, 620-670nm
 6  // NIR is sur_refl_b02, 841-876nm
 7
 8  var img = ee.Image('MOD09GA/MOD09GA_005_2012_03_09');
 9  var ndvi = img.normalizedDifference(['sur_refl_b02', 'sur_refl_b01']);
10  var palette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',
11                 '74A901', '66A000', '529400', '3E8601', '207401', '056201',
12                 '004C00', '023B01', '012E01', '011D01', '011301'];
13
14  Map.setCenter(-94.84497, 39.01918, 8);
15  Map.addLayer(img.select(['sur_refl_b01', 'sur_refl_b04', 'sur_refl_b03']),
16           {gain: '0.1, 0.1, 0.1'}, 'MODIS bands 1/4/3');
17  Map.addLayer(ndvi, {min: 0, max: 1, palette: palette}, 'NDVI');
18
19
```

5. **Read** the Normalized Difference script, line by line (or statement by statement), to see what it is doing:
   i. Lines 1 to 6 are comment lines the developer included to describe the script. Line comments are designated with the //, the double slashes at the beginning of the line. Comments are ignored by the Code Editor when the script executes.
   ii. Line 8 accomplishes two things. It declares a variable, called *img*. It then assigns a value to this variable. The value is a MODIS image ee.Image('MOD09GA/MOD09GA_005_2012_03_09').
   iii. Line 9 does several things. It declares and assigns a value to a variable, called *ndvi*. It also calls the Earth Engine NormalizedDifference method and applies it to the variable "img" defined in the previous line. The bands "sur_refl_b02" and "sur_refl_b01" are specified as inputs to that calculation (arguments to the method). These two bands are the NIR and Red MODIS bands, so the result of the calculation is a Normalized Difference Vegetation Index (NDVI) image. This calculated NDVI image is what is being assigned to the *ndvi* variable. Specifically, it's an image that represents the computation:

      (sur_refl_b02 - sur_refl_b01) / (sur_refl_b02 + sur_refl_b01).



```
1  // NormalizedDifference example.
2  //
3  // Compute Normalized Difference Vegetation Index over MOD09GA product.
4  // NDVI = (NIR - RED) / (NIR + RED), where
5  // RED is sur_refl_b01, 620-670nm
6  // NIR is sur_refl_b02, 841-876nm
7
8  var img = ee.Image('MOD09GA/MOD09GA_005_2012_03_09');
9  var ndvi = img.normalizedDifference(['sur_refl_b02', 'sur_refl_b01']);
10 var palette = ['FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718',
11                '74A901', '66A000', '529400', '3E8601', '207401', '056201',
12                '004C00', '023B01', '012E01', '011D01', '011301'];
13
14 Map.setCenter(-94.84497, 39.01918, 8);
15 Map.addLayer(img.select(['sur_refl_b01', 'sur_refl_b04', 'sur_refl_b03']),
16          {gain: '0.1, 0.1, 0.1'}, 'MODIS bands 1/4/3');
17 Map.addLayer(ndvi, {min: 0, max: 1, palette: palette}, 'NDVI');
18
19
```

normalizedDifference(*bandNames*)

Computes a normalized difference between two bands in an image. If the bands to use are not specified, uses the first two bands.

Arguments:

- **this:input (Image):**
  The input image.

- ***bandNames (List, default: null):***
  A list of names specifying the bands to use. If not specified, the first and second bands are used.

Returns: Image

iv. Lines 10-12 declare a variable, palette, and assign to it an array that specifies a palette of hexadecimal color codes for displaying the resulting NDVI image. The hexadecimal colors range from white (FFFFFF) to browns (e.g., CE7E45) to yellows (e.g., FCD163) to greens (e.g., 529400) to very dark (011301).

***Note:** You can read more about hexadecimal color codes here http://www.colorhexa.com/.*

v. Line 14 centers the map to the area of interest. The arguments, the values inside the brackets, are the longitude and latitude values for Kansas City, USA; the third value sets the zoom level.

vi. Lines 15-17 add data to the map output window (lower panel). Two images are displayed - the *img* variable, which points to the original MODIS image, and the *ndvi* variable, which points to the normalized difference image created in line 9.

6. Click the **Run** button in the upper-right of the code editor to run the Normalized Difference script.

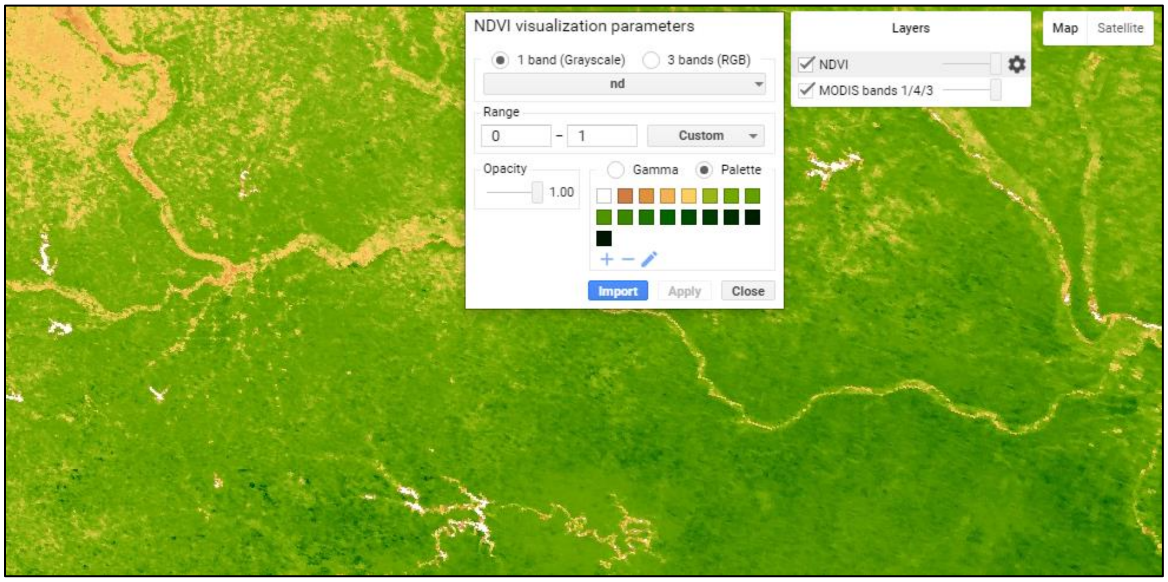
   i. You should see a MODIS image and the resulting NDVI image appear in the map output window at the bottom of your screen.

7. Visually examine the results in the map output window using the map viewer tools.

   i. Click or mouse-hover on the **Layers** button in the upper right hand corner of the Map output panel at the bottom of your screen (shown in the following graphic).

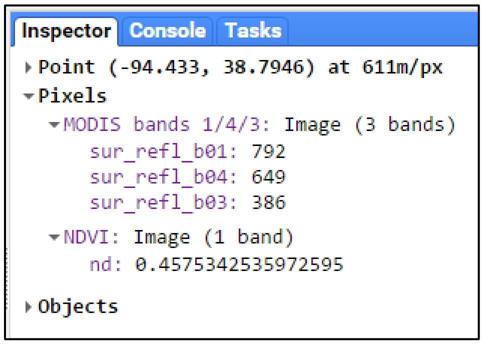
   ii. Toggle the NDVI layer on and off by **unchecking and checking** the box next to the NDVI Layer.

iii. Click and drag the slider-bar back and forth to adjust the transparency of the NDVI layer and view the MODIS image beneath the NDVI image (see following image, with visualization parameter box).



8. Use the **Inspector** Panel to explore the values in the resulting NDVI image.
   i. Click on the Inspector tab in the upper right-hand panel.
      (a) Hover your cursor over the map. Notice that your cursor has become a cross.

   ii. **Click anywhere on the map** and observe the values that appear in the window under the Inspector tab.
      (a) These are the pixel values at this location for:
         (i) MODIS band values for the displayed bands appear under the MODIS image name.
         (ii) The computed NDVI values.



```
▸Point (-94.433, 38.7946) at 611m/px
▾Pixels
  ▾MODIS bands 1/4/3: Image (3 bands)
     sur_refl_b01: 792
     sur_refl_b04: 649
     sur_refl_b03: 386
  ▾NDVI: Image (1 band)
     nd: 0.4575342535972595
▸Objects
```

Part 4: (*optional*) Explore Data Available in Earth Engine

A. The Earth Engine Data Catalog

1. In a web browser, such as Google Chrome, open the Google Earth Engine homepage: https://earthengine.google.com/.
2. Click on **Datasets** in the upper right corner. This will give you a quick overview of some of the data that is available in Earth Engine. Take a moment to read through the information on imagery, geophysical data, climate and weather, and demographic data.
3. Towards the top of the page you can click **View all Datasets** to open a new window where you can search for other data in the catalog that you might be interested in using for your own scripts.

Congratulations! You've completed this exercise and learned some basics about the code editor and Earth Engine datasets. In the next exercise you'll learn to use more Earth Engine methods.