[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

PROCESSING **FILTER** [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)

A filter is an Earth Engine parameter object that represents a procedure by which a new image collection or feature collection is generated by copying only those images or features of a specified collection that satisfy specified conditions. Filters can be processed by using operations of the types listed below, which vary according to the nature of that processing. Each operation name is linked to a separate page describing that operation.

**CREATING** FILTERS

TO TEST FOR **RELATIVE MAGNITUDE**  [ee.Filter.eq](#eq_) [ee.Filter.neq](#eq_)

[ee.Filter.gt](#eq_) [ee.Filter.gte](#eq_)

[ee.Filter.lt](#eq_) [ee.Filter.lte](#eq_)

[ee.Filter.equals](#equals_) [ee.Filter.notequals](#equals_)

[ee.Filter.greaterThan](#equals_) [ee.Filter.greaterThanOrEquals](#equals_)

[ee.Filter.lessThan](#equals_) [ee.Filter.lessThanOrEquals](#equals_)

[ee.Filter.maxDifference](#maxDifference)

TO TEST FOR **FACTUAL COINCIDENCE** [ee.Filter.stringContains](#stringContains_) [ee.Filter.StartsWith](#stringContains_) [ee.FilterEndsWith](#stringContains_)

[ee.Filter.rangeContains](#rangeContains) [ee.Filter.listContains](#listContains) [ee.Filter.inList](#inList)

TO TEST FOR **TEMPORAL COINCIDENCE** [ee.Filter.calendarRange](#calendarRange) [ee.Filter.dateRangeContains](#dataRangeContains) [ee.Filter.dayOfYear](#dayOfYear)

TO TEST FOR **SPATIAL COINCIDENCE**  [ee.Filter.bounds](#bounds) [ee.Filter.intersects](#intersects)

[ee.Filter.disjoint](#disjoint) [ee.Filter.withinDistance](#withinDistance)

**TRANSFORMING** FILTERS [ee.Filter.and](#and_) [ee.Filter( )](#and_)

[ee.Filter.or](#or) [ee.Filter.not](#not)

**DOCUMENTING** FILTERS filter.getInfo ee.Algorithms.Describe(filter)

[filter.toString](#toString) [filter.serialize](#serialize)

**PRESENTING** FILTERS

IN **PRINT** print(filter) console.log(filter)

alert(filter) confirm(filter)

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **RELATIVE MAGNITUDE**

ee.Filter.eq , .neq , .gt , .gte , .lt , and .lte create new filters, each of which tests whether the value of a specified property

is equal to, not equal to, greater than, greater than or equal to, less than. or

less than or equal to a specified value

newFilter = ee.Filter.eq( testProperty, testValue )

or .neq

The specified value

or .gt

The new filter

or .gte

or .lt

The name of the specified property, given as a string

or .lte

var AllFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

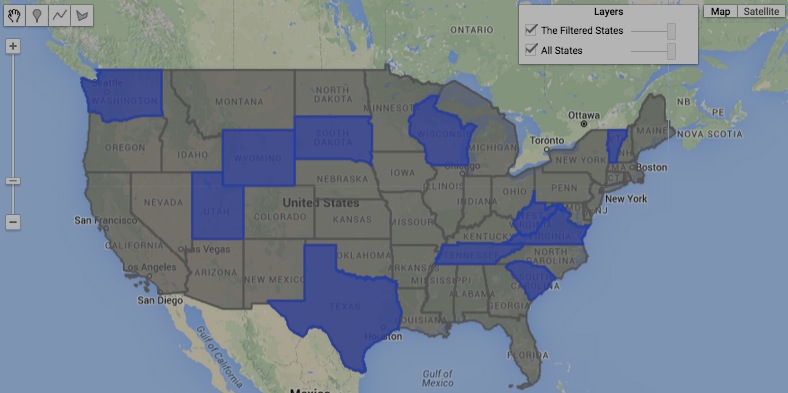
var TheFILTER = ee.Filter.gt( 'Name', 'Rhode Island' );

var SomeFEATURES = AllFEATURES.filter( TheFILTER );

Map.setCenter( -95.9766, 39.3683, 4 );

Map.addLayer ( AllFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( SomeFEATURES, {color:'4444ee'}, 'The Filtered States');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **RELATIVE MAGNITUDE**

ee.Filter.equals , .notEquals , create new filters, each of which tests whether the value

.greaterThan , .greaterThanOrEquals ,

of a specified property is equal to, not equal to, greater than,

greater than or equal to, less than. or less than or equal to

a specified value

.lessThan , and .lessThanOrEquals

newFilter = ee.Filter.equals ( testProperty, testValue ) or ( testValue, testProperty )

The name of the specified property, given as a string

The specified value

or .notEquals

or .greaterThan

or .greaterThanOrEquals

The new filter

or .lessThan

or .lessThanOrEquals

var AllFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

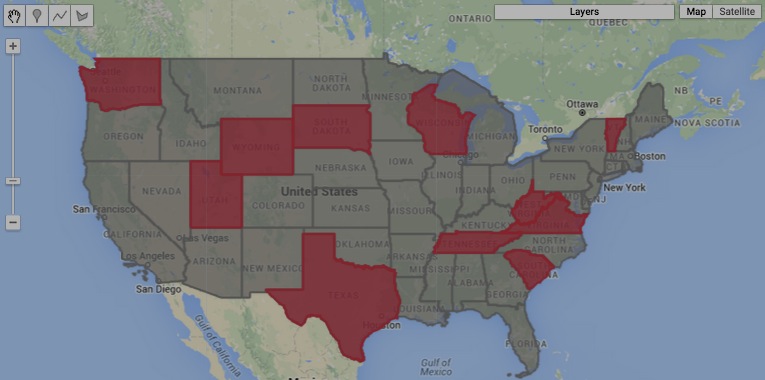
var TheFILTER = ee.Filter.greaterThan( 'Name', 'Rhode Island' );

var SomeFEATURES = AllFEATURES.filter( TheFILTER );

Map.setCenter( -95.9766, 39.3683, 4 );

Map.addLayer ( AllFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( SomeFEATURES, {color:'cc1133'}, 'The Filtered States');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **RELATIVE MAGNITUDE**

ee.Filter.maxDifference create new filters, each of which tests whether the value of a specified property

differs from a specified value by less than a specified amount.

newFilter = ee.Filter.maxDifference( difference, testProp, testValue ) or ( difference, testValue, testProp )

The specified amount, given as a float

The specified value

The new filter

The name of the specified property, given as a string

var AllFEATURES = ee.FeatureCollection( 'ft:1xa2PvKTf7ynyAAEXEeHoltriaHFkyFJpvd74BLc6' ); // CT Census Tracts

var TheFILTER = ee.Filter.maxDifference( 0.001, 'AREA', 0.009 );

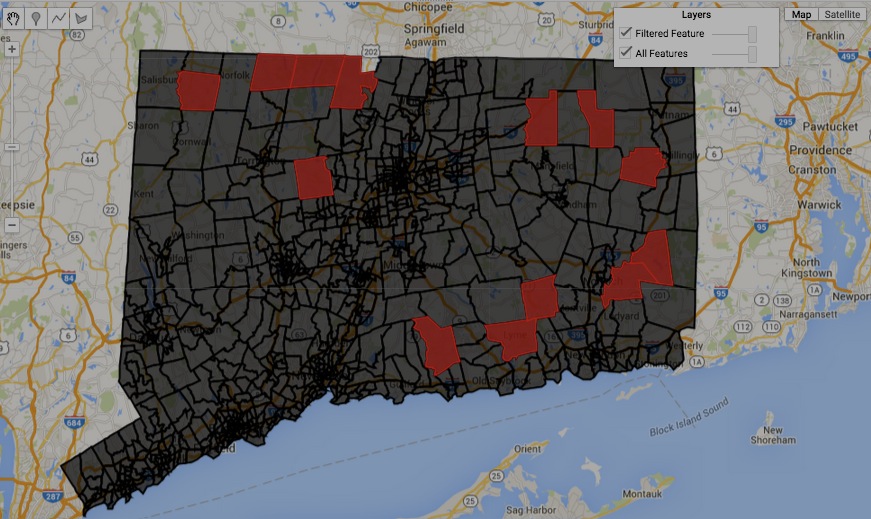
var SomeFEATURES = AllFEATURES.filter( TheFILTER );

print( AllFEATURES, SomeFEATURES );

Map.centerObject( AllFEATURES, 9 );

Map.addLayer( AllFEATURES, null, 'All Features' );

Map.addLayer( SomeFEATURES, {color:"ee0000"}, 'Filtered Feature' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **RELATIVE MAGNITUDE**

ee.Filter.stringContains, .stringStartsWith, and .stringEndsWith

create new filters, each of which tests whether the value of a specified string property contains, starts with, or ends with a specified substring.

newFilter = ee.Filter.stringContains( testProperty, testString ) or ( testString, testProperty )

or .stringStartsWith

The specified string

or .stringEndsWith

The new filter

The name of the specified property, given as a string

var ThisFILTER = ee.Filter.stringStartsWith( 'Name', 'A');

var ThatFILTER = ee.Filter.stringEndsWith( 'Name', 'a');

var AllFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var TheseFEATURES = AllFEATURES.filter( ThisFILTER );

var ThoseFEATURES = AllFEATURES.filter( ThatFILTER );

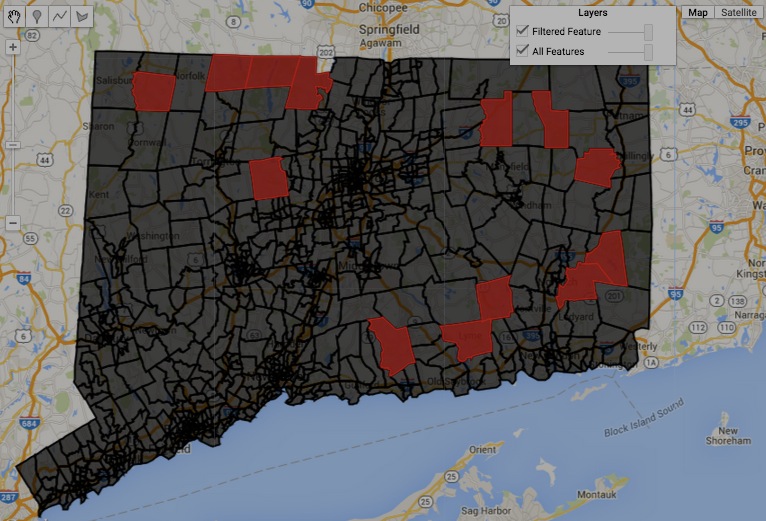
print( TheseFEATURES, ThoseFEATURES );

Map.setCenter( -95.9766, 39.3683, 4 );

Map.addLayer ( AllFEATURES, {color:'aaaaaa'}, 'All States' );

Map.addLayer ( TheseFEATURES, {color:'ff2222'}, 'The Initial "A" States');

Map.addLayer ( ThoseFEATURES, {color:'aaaaff'}, 'The Final "a" States' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **RELATIVE MAGNITUDE**

ee.Filter.rangeContains creates a new filter that tests whether the value in a specified field lies within a specified range of value.

newFilter = ee.Filter.rangeContains( fieldName, minimumValue, maximumValue )

The new filter

The name of the specified

field, given as a string

The minimum value to be included

within the specified range

The maximum value to be included

within the specified range

var TheFILTER = ee.Filter.rangeContains( 'Name', 'N','P');

var AllFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

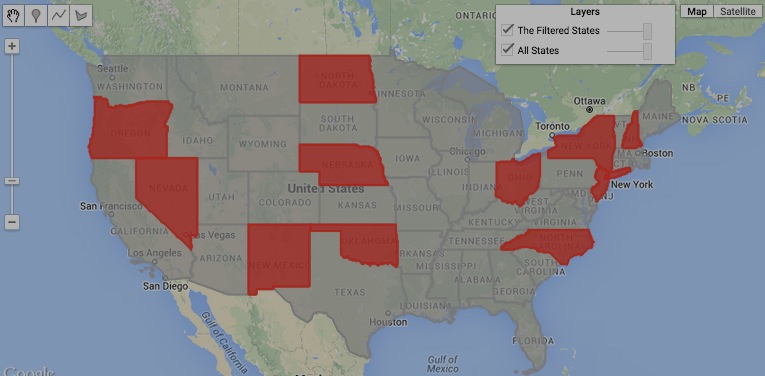
var SomeFEATURES = AllFEATURES.filter( TheFILTER );

print( SomeFEATURES );

Map.setCenter( -95.9766, 39.3683, 4 );

Map.addLayer ( AllFEATURES, {color:'aaaaaa'}, 'All States' );

Map.addLayer ( SomeFEATURES, {color:'ff0000'}, 'The Filtered States');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **FACTUAL COINCIDENCE**

ee.Filter.listContains creates a new filter that tests whether or not a specified field contains a specified value.

newFilter = ee.Filter.listContains( ThisField, ThatValue, ThatField, ThisValue)

The name of the specified field, given as a

string. Mutually exclusive with **ThisValue**.

The new

filter

The specified value. Mutually

exclusive with **ThisField**.

The specified value. Mutually

exclusive with **ThatField**.

The name of the specified field, given as a

string. Mutually exclusive with **ThatValue**

var UtahGEOMETRY = ee.Geometry.Polygon( [[-114.05,37],[-109.05,37],[-109.05,41],[-111.05,41],[-111.05,42],[-114.05,42]] );

var ColoGEOMETRY = ee.Geometry.Polygon( [[-109.05,41],[-109.05,37],[-102.05,37],[-102.05,41] ] );

var WyomGEOMETRY = ee.Geometry.Polygon([[-111.05,41],[-111.05,45],[-104.10,45],[-104.10,41] ] );

var UtahFEATURE = ee.Feature( UtahGEOMETRY, {name:'Utah', letters:['U','T','A','H' ]} );

var ColoFEATURE = ee.Feature( ColoGEOMETRY, {name:'Colorado', letters:['C','O','L','O','R','A','D','O']} );

var WyomFEATURE = ee.Feature( WyomGEOMETRY, {name:'Wyoming', letters:['W','Y','O','M','I','N','G' ]} );

var AllFEATURES = ee.FeatureCollection( [UtahFEATURE,ColoFEATURE,WyomFEATURE], 'geometry' );

var TheFILTER = ee.Filter.listContains( 'letters', 'O' );

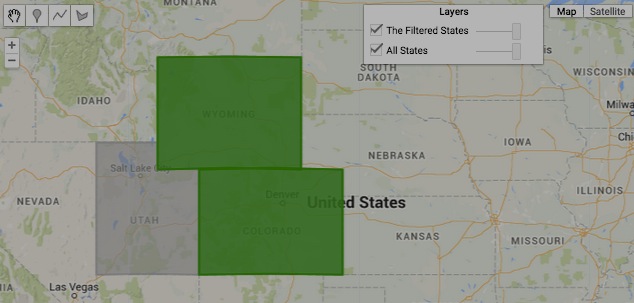
var FilteredFEATURES = AllFEATURES.filter( TheFILTER );

print( AllFEATURES, FilteredFEATURES );

Map.centerObject( AllFEATURES, 5 );

Map.addLayer ( AllFEATURES, {color:'aaaaaa'}, 'All States' );

Map.addLayer ( FilteredFEATURES, {color:'009900'}, 'The Filtered States');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **FACTUAL COINCIDENCE**

ee.Filter.inList creates a new filter that tests whether or not the value of a specified property matches any of a specified set of values.

newFilter = ee.Filter.inList( testProperty, testValue ) or ( testValue, testProperty )

The specified value or list of values

The new filter

The specified property

var TheLIST = ee.List( ['Connecticut','Maine','Massachusetts','New Hampshire','Rhode Island','Vermont'] );

var TheFILTER = ee.Filter.inList( 'Name', TheLIST);

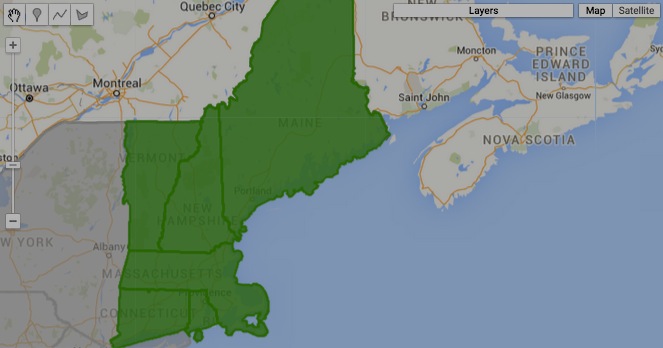
var AllFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var SomeFEATURES = AllFEATURES.filter( TheFILTER );

Map.setCenter( -70.488, 44.088, 5 );

Map.addLayer ( AllFEATURES, {color:'aaaaaa'}, 'All States' );

Map.addLayer ( SomeFEATURES, {color:'009900'}, 'The Filtered States');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **FACTUAL COINCIDENCE**

ee.Filter.calendarRange creates a new filter that tests whether the value in specified calendar field falls within a time period. This period

begins at a specified starting time and ends at a specified stopping time (inclusively) unless the specified starting time

is later than the specified stopping time. In the case, the period includes all time before (and including) the specified stopping time or after (and including) the specified starting time.

newFilter = ee.Filter.calendarRange( startingTime*, stoppingTime, calendarField* )

The specified calendar field, given as

one of the following strings: **'year'**,

**'month'**, **'hour'**, **'minute'**, **'day\_of\_year**', **'day\_of\_month**', or **'day\_of\_wee**k'.

Default: **'day\_of\_year**.'

The specified stopping time, given as an

integer identifying the month (1–12),

day of the year (1-365), day of the month

(1-31), or day of the week (1-7) in

[Universal Coordinated Time](https://en.wikipedia.org/wiki/ISO_8601) (UTC).

Default: **startingTime**

The specified starting time, given as

an integer identifying the month (1–12),

day of the year (1-365), day of the month

(1-31), or day of the week (1-7) in

[Universal Coordinated Time](https://en.wikipedia.org/wiki/ISO_8601) (UTC).

The new

filter

var TheGEOMETRY = ee.Geometry.Point( -111.83533, 34.57499 ); // Arizona

var BoundedIMAGES = ee.ImageCollection('LE7\_L1T').filterBounds( TheGEOMETRY );

var DatedIMAGES = BoundedIMAGES.filterDate( '2000-01-01', '2002-01-01' );

var TheFILTER = ee.Filter.calendarRange( 160,200 );

var SeasonalIMAGES = DatedIMAGES.filter( TheFILTER );

var ImageLIST = SeasonalIMAGES.toList( 5,0 );

print( '348 Images by Location', BoundedIMAGES );

print( '044 Images by Location and Date', DatedIMAGES );

print( '005 Images by Location, Date, and DayOfYear', SeasonalIMAGES );

Map.setCenter( -111.83533, 34.57499, 8 );

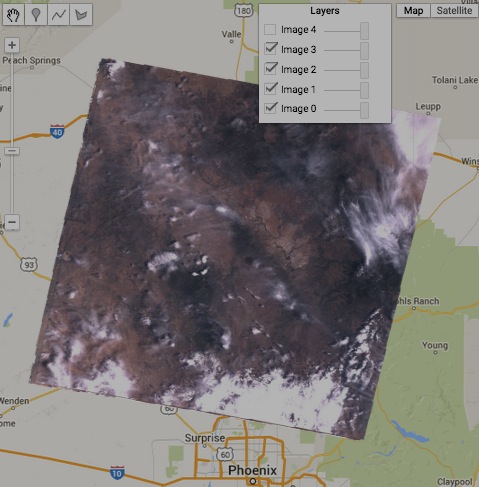
Map.addLayer( ee.Image( ImageLIST.get(0) ), {bands:['B3,B2,B1']}, 'Image 0');

Map.addLayer( ee.Image( ImageLIST.get(1) ), {bands:['B3,B2,B1']}, 'Image 1');

Map.addLayer( ee.Image( ImageLIST.get(2) ), {bands:['B3,B2,B1']}, 'Image 2');

Map.addLayer( ee.Image( ImageLIST.get(3) ), {bands:['B3,B2,B1']}, 'Image 3');

Map.addLayer( ee.Image( ImageLIST.get(4) ), {bands:['B3,B2,B1']}, 'Image 4');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **TEMPORAL COINCIDENCE**

ee.Filter.dateRangeContains creates a new filter that tests whether the value in specified dateRange contains a specified date.

newFilter = ee.Filter.dateRangeContains( ThisField, ThatDate, ThatField, ThisDate)

The specified dateRange. Mutually

exclusive with **ThisField**.

The name of a dateRange field, given

as a string. Mutually exclusive with **ThisDate**.

The new

filter

The specified date. Mutually

exclusive with **ThatField**.

The name of a date field, given as a

string. Mutually exclusive with **ThatDate**

var TheGEOMETRY = ee.Geometry.Point( -111.83533, 34.57499 ); // Arizona

var BoundedIMAGES = ee.ImageCollection('LE7\_L1T').filterBounds( TheGEOMETRY );

var TheDATERANGE = ee.DateRange( '2000-01-01', '2000-03-01' );

var TheFILTER = ee.Filter.dateRangeContains( null, null, 'DATE\_ACQUIRED', TheDATERANGE );

var DatedIMAGES = BoundedIMAGES.filter( TheFILTER );

var ImageLIST = DatedIMAGES.toList( 5,0 );

print( 'Images by Location', BoundedIMAGES );

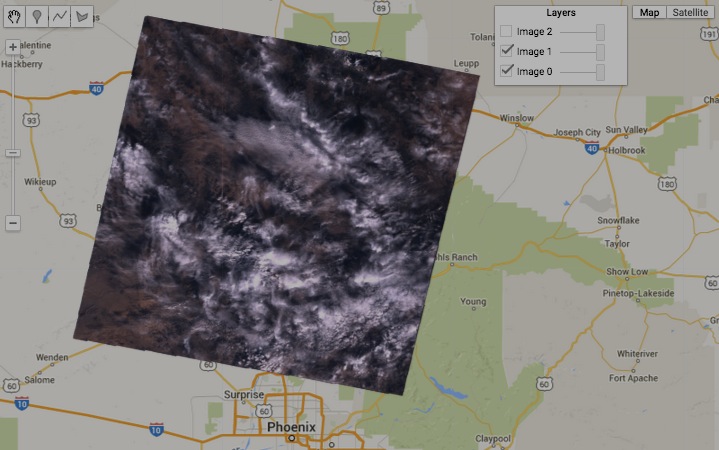
print( 'Images by Location and Date', DatedIMAGES );

Map.setCenter( -111.83533, 34.57499, 8 );

Map.addLayer( ee.Image( ImageLIST.get(0) ), {bands:['B3,B2,B1']}, 'Image 0');

Map.addLayer( ee.Image( ImageLIST.get(1) ), {bands:['B3,B2,B1']}, 'Image 1');

Map.addLayer( ee.Image( ImageLIST.get(2) ), {bands:['B3,B2,B1']}, 'Image 2');

****

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **TEMPORAL COINCIDENCE**

ee.Filter.dayOfYear creates a new filter that tests whether an object's timestamp falls within the given day-of-year range.

newFilter = ee.Filter.dayOfYear( startingTime, stoppingTime )

The specified stopping time, given as an integer identifying the day of the year

(1-365) in Universal Coordinated Time (UTC). Default: **startingTime**

The specified starting time, given as an integer identifying the

day of the year (1-365) in Universal Coordinated Time (UTC).

The new

filter

var JanuaryFILTER = ee.Filter.dayOfYear( 1, 30 );

var FebruaryFILTER = ee.Filter.dayOfYear(31, 46 );

var TheGEOMETRY = ee.Geometry.Point( -111.83533, 34.57499 ); // Arizona

var BoundedIMAGES = ee.ImageCollection('LE7\_L1T').filterBounds( TheGEOMETRY );

var DatedIMAGES = BoundedIMAGES.filterDate( '2000-01-01', '2002-01-01' );

var JanuaryIMAGES = DatedIMAGES.filter( JanuaryFILTER);

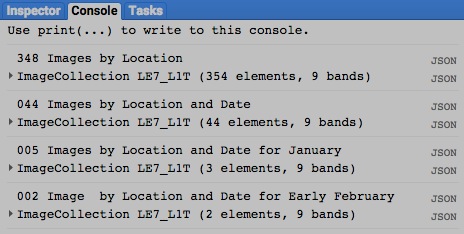
var FebruaryIMAGES = DatedIMAGES.filter( FebruaryFILTER);

print( '348 Images by Location', BoundedIMAGES );

print( '044 Images by Location and Date', DatedIMAGES );

print( '005 Images by Location and Date for January', JanuaryIMAGES );

print( '002 Image by Location and Date for Early February', FebruaryIMAGES );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **SPATIAL COINCIDENCE**

ee.Filter.bounds creates a new filter that tests whether the geometry of a feature or image touches that of a specified spatial object.

newFilter = ee.Filter.bounds( spatialObject*, errorMargin* )

A maximum amount of error to be tolerated, given as an ErrorMargin

or a Number referring to spherical meters. Default: 0.

The specified spatial object, given as a geometry.

( Features and feature collections should also work

but seem not to).

The new

filter

var AllStateFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var AllRegionCOLLECTION = ee.FeatureCollection('ft:1Ec8IWsP8asxN-ywSqgXWMuBaxI6pPaeh6hC64lA');

var OneRegionCOLLECTION = AllRegionCOLLECTION.filter(ee.Filter().eq('ECO\_NAME', 'Southeastern mixed forests'));

var OneRegionELEMENT = OneRegionCOLLECTION.first( );

var OneRegionFEATURE = ee.Feature( OneRegionELEMENT );

var OneRegionGEOMETRY = OneRegionFEATURE.geometry();

var TheFILTER = ee.Filter.bounds( OneRegionGEOMETRY );

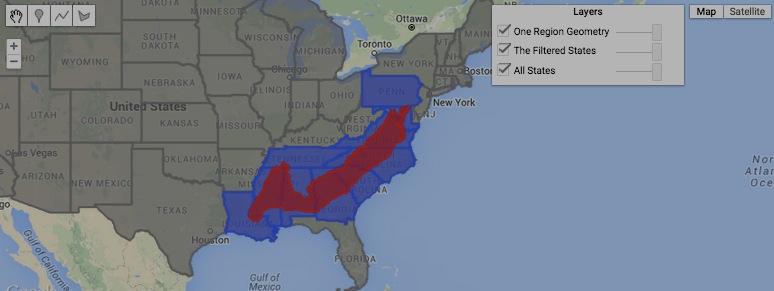
var SomeStateFEATURES = AllStateFEATURES.filter( TheFILTER );

Map.centerObject( OneRegionGEOMETRY, 4 );

Map.addLayer ( AllStateFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( SomeStateFEATURES, {color:'4444ee'}, 'The Filtered States');

Map.addLayer ( OneRegionGEOMETRY, {color:'dd0000'}, 'One Region Geometry' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **SPATIAL COINCIDENCE**

ee.Filter.intersects creates a new filter that tests whether the geometry of a feature or image touches that of a specified spatial object.

newFilter = ee.Filter.intersects( 1stField, 2ndGeometry, 2ndField, 1stGeometry*, errorMargin* )

The first specified geometry, given as

a string naming a a geometry field.

Mutually exclusive with **1stField**

The second specified geometry.

Mutually exclusive with **2ndField**.

The new filter

The first specified geometry, given as a string naming a

geometry field. Mutually exclusive with **1stGeometry**.

The second specified geometry, given as a string naming a

geometry field. Mutually exclusive with **2ndGeometry**.

A maximum amount of error to be tolerated, given as an ErrorMargin

or a Number referring to spherical meters. Default: 0.

var AllStateFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var AllRegionCOLLECTION = ee.FeatureCollection('ft:1Ec8IWsP8asxN-ywSqgXWMuBaxI6pPaeh6hC64lA'); // Ecoregions

var OneRegionCOLLECTION = AllRegionCOLLECTION.filter(ee.Filter().eq('ECO\_NAME', 'Southeastern mixed forests'));

var OneRegionELEMENT = OneRegionCOLLECTION.first( );

var OneRegionFEATURE = ee.Feature( OneRegionELEMENT );

var OneRegionGEOMETRY = OneRegionFEATURE.geometry();

var TheFILTER = ee.Filter.intersects( '.geo', OneRegionGEOMETRY );

var SomeStateFEATURES = AllStateFEATURES.filter( TheFILTER );

print( AllStateFEATURES.limit(3) );

//print( SomeStateFEATURES.limit(3) );

Map.centerObject( OneRegionGEOMETRY, 4 );

Map.addLayer ( AllStateFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( SomeStateFEATURES, {color:'4444ee'}, 'The Filtered States');

Map.addLayer ( OneRegionGEOMETRY, {color:'dd0000'}, 'One Region Geometry' );

EXAMPLE NOT YET BEHAVING

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **SPATIAL COINCIDENCE**

ee.Filter.disjoint creates a new filter that tests whether the geometry of a feature or image does not touch that of a specified spatial object.

newFilter = ee.Filter.intersects( spatialObject*, errorMargin* )

A maximum amount of error to be tolerated, given as an ErrorMargin

or a Number referring to spherical meters. Default: 0.

The specified spatial object, given as

a Geometry, a Feature, or a Feature Collection.

The new

filter

var AllStateFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var AllRegionCOLLECTION = ee.FeatureCollection('ft:1Ec8IWsP8asxN-ywSqgXWMuBaxI6pPaeh6hC64lA'); // Ecoregions

var OneRegionCOLLECTION = AllRegionCOLLECTION.filter(ee.Filter().eq('ECO\_NAME', 'Southeastern mixed forests'));

var OneRegionELEMENT = OneRegionCOLLECTION.first( );

var OneRegionFEATURE = ee.Feature( OneRegionELEMENT );

var OneRegionGEOMETRY = OneRegionFEATURE.geometry();

var TheFILTER = ee.Filter.disjoint( '.geo', OneRegionGEOMETRY );

var SomeStateFEATURES = AllStateFEATURES.filter( TheFILTER );

print( AllStateFEATURES.limit(3) );

//print( SomeStateFEATURES.limit(3) );

Map.centerObject( OneRegionGEOMETRY, 4 );

Map.addLayer ( AllStateFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( SomeStateFEATURES, {color:'4444ee'}, 'The Filtered States');

Map.addLayer ( OneRegionGEOMETRY, {color:'dd0000'}, 'One Region Geometry' );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**CREATING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) TO TEST FOR **SPATIAL COINCIDENCE**

ee.Filter.withinDistance creates a new filter that tests whether the geometry of one specified geometry

comes to within a specified maximum distance a second specified geometry.

newFilter = ee.Filter.withinDistance( distance, 1stField, 2ndGeometry, 2ndField, 1stGeometry*, errorMargin* )

The specified distance in meters,

given as a floating-point number

The second specified geometry.

Mutually exclusive with **2ndField**.

The first specified geometry, given as

a string naming a a geometry field.

Mutually exclusive with **1stField**

The new filter

The second specified geometry, given as a string naming a

geometry field. Mutually exclusive with **2ndGeometry**

The first specified geometry, given as a string naming a

geometry field. Mutually exclusive with **1stGeometry**.

A maximum amount of error to be tolerated, given as an ErrorMargin

or a Number referring to spherical meters. Default: 0.

var AllStateFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var AllRegionCOLLECTION = ee.FeatureCollection('ft:1Ec8IWsP8asxN-ywSqgXWMuBaxI6pPaeh6hC64lA'); // Ecoregions

var OneRegionCOLLECTION = AllRegionCOLLECTION.filter(ee.Filter().eq('ECO\_NAME', 'Southeastern mixed forests'));

var OneRegionELEMENT = OneRegionCOLLECTION.first( );

var OneRegionFEATURE = ee.Feature( OneRegionELEMENT );

var OneRegionGEOMETRY = OneRegionFEATURE.geometry();

var TheFILTER = ee.Filter.withinDistance( 5000, '.geo', OneRegionGEOMETRY );

var SomeStateFEATURES = AllStateFEATURES.filter( TheFILTER );

print( AllStateFEATURES.limit(3) );

//print( SomeStateFEATURES.limit(3) );

Map.centerObject( OneRegionGEOMETRY, 4 );

Map.addLayer ( AllStateFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( SomeStateFEATURES, {color:'4444ee'}, 'The Filtered States');

Map.addLayer ( OneRegionGEOMETRY, {color:'dd0000'}, 'One Region Geometry' );

EXAMPLE EXCEEDING TILE LIMIT

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**TRANSFORMING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx)

ee.Filter.and and ee.Filter and each creates a new filter that tests whether all of a specified list of filters is satisfied. The same effect can also be achieved by

“chaining” filters as illustrated here: var collection = ee.ImageCollection('L7\_L1T').filterDate('2000-04-01', '2000=07-01').filterBounds(polygon);

newFilter = ee.Filter or ee.Filter.and( filterList)

The specified list of filters

The new filter

var AllFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var FirstLetterFILTER = ee.Filter.stringStartsWith( 'Name', 'A');

var FinalLetterFILTER = ee.Filter.stringEndsWith( 'Name', 'a');

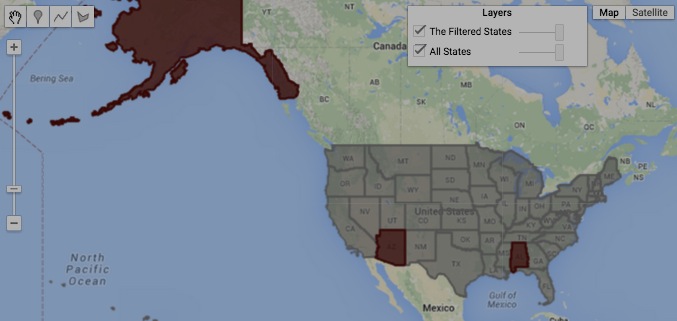
var CombinedFILTER = ee.Filter( FirstLetterFILTER, FinalLetterFILTER );

var FilteredFEATURES = AllFEATURES.filter( CombinedFILTER );

Map.setCenter( -95.9766, 39.3683, 3 );

Map.addLayer ( AllFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( FilteredFEATURES, {color:'550000'}, 'The Filtered States');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**TRANSFORMING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx)

ee.Filter.or each creates a new filter that tests whether any of a specified list of filters is satisfied.

newFilter = ee.Filter.or( filterList)

The specified list of filters

The new filter

var AllFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var FirstLetterFILTER = ee.Filter.stringStartsWith( 'Name', 'A');

var FinalLetterFILTER = ee.Filter.stringEndsWith( 'Name', 'a');

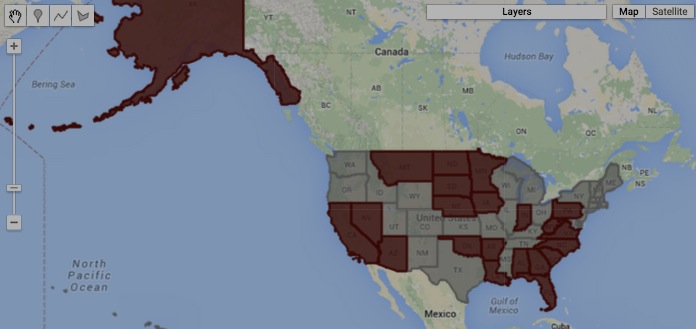
var CombinedFILTER = ee.Filter.or( FirstLetterFILTER, FinalLetterFILTER );

var FilteredFEATURES = AllFEATURES.filter( CombinedFILTER );

Map.setCenter( -95.9766, 39.3683, 3 );

Map.addLayer ( AllFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( FilteredFEATURES, {color:'550000'}, 'The Filtered States');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**TRANSFORMING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx)

ee.Filter.not each creates a new filter that tests whether a specified filter is unsatisfied.

newFilter = ee.Filter.not( oldFilter)

The specified filter

The new filter

var AllFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var FirstLetterFILTER = ee.Filter.stringStartsWith( 'Name', 'A');

var FinalLetterFILTER = ee.Filter.stringEndsWith( 'Name', 'a');

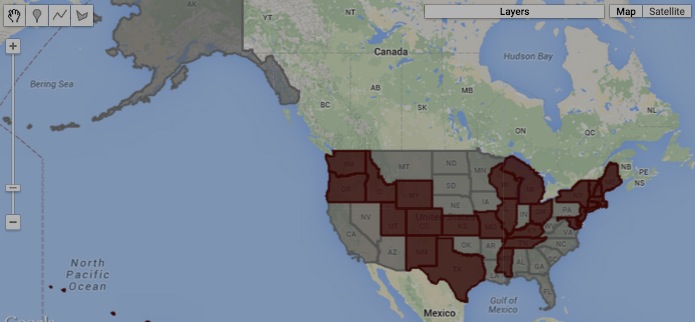
var CombinedFILTER = ee.Filter.or( FirstLetterFILTER, FinalLetterFILTER );

var CombinedFILTER = CombinedFILTER.not();

var FilteredFEATURES = AllFEATURES.filter( CombinedFILTER );

Map.setCenter( -95.9766, 39.3683, 3 );

Map.addLayer ( AllFEATURES, {color:'777777'}, 'All States' );

Map.addLayer ( F iltere dFEATURES, {color:'550000'}, 'The Filtered States');

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**DOCUMENTING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx)

ee.Algorithms.Describe and filter.getInfo

each creates a JSON-compatible text object representing a specified filter.

newObject = ee.Algorithms.Describe( oldFilter )

and oldFilter.getInfo( )

The specified filter

The new object

var TheFILTER = ee.Filter.eq( 'abc' , 'def' );

print( 'From print:', TheFILTER );

print( 'From ee.Algorithms.Describe( ):', ee.Algorithms.Describe( TheFILTER ) );

print( 'From getInfo( ):', TheFILTER.getInfo( ) );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**DOCUMENTING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx)

filter.toString and .serialize each creates a new string presenting information on a specified filter.

newString = oldFilter.toString( )

and oldFilter.serialize( )

The specified filter

The new string

var TheFILTER = ee.Filter.eq( 'abc' , 'def' );

print( 'From print:', TheFILTER );

print( 'From toString( ):', TheFILTER.toString( ) );

print( 'From serialize( ):', TheFILTER.serialize( ) );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) IN **PRINT**

print ( filter ) and console.log ( filter ) present JSON-formatted text renditions of a specified filter in the console.

print( oldFilter ) or console.log( oldFilter )

The specified filter

var TheFILTER = ee.Filter.eq( 'abc' , 'def' );

print( 'From print:', TheFILTER );

console.log( 'From console.log:', TheFILTER );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [FILTER](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx) IN **PRINT**

alert ( filter ) and confirm ( filter ) present JSON-formatted text renditions of a specified

filter in a pop-up message box.

alert( oldFilter ) or confirm( oldFilter )

The specified filter

var TheFILTER = ee.Filter.eq( 'abc' , 'def' );

alert( TheFILTER );

confirm( TheFILTER );