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

A join is an Earth Engine parameter object that represents a procedure by which two collections of corresponding elements are compared to one another and a new element is (potentially) generated from each corresponding pair. Joins 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** JOINS [ee.Join.simple](#simple) [ee.Join.inverted](#inverted) [ee.Join.inner](#inner)

[ee.Join.saveAll](#saveAll) [ee.Join.saveBest](#saveBest) [ee.Join.saveFirst](#saveFirst)

**APPLYING** JOINS [join.apply](#apply)

**DOCUMENTING** JOINS join.getInfo ee.Algorithms.Describe(join)

join.toString join.serialize

**PRESENTING** JOINS

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

alert(join) confirm(join)

[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** [JOIN](#_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.Join.simple creates a new join that retains only those elements of a “primary” (image or feature) collection that match an element of a “secondary”

(image or feature) collection. Once created, this new join can be applied to specified data by using a **join.apply** operation.

newJoin = ee.Join.simple( )

The new join

var CityFEATURES = ee.FeatureCollection( 'ft:1G3RZbWoTiCiYv\_LEwc7xKZq8aYoPZlL5\_KuVhyDM' ); // U.S. Cities

var CityFEATURES = CityFEATURES.filterMetadata('city\_name','contains','Pa.'); // Pennsylvania Cities

var CountyFEATURES = ee.FeatureCollection('ft:1S4EB6319wWW2sWQDPhDvmSBIVrD3iEmCLYB7nMM'); // U.S. Counties

var CountyFEATURES = CountyFEATURES.filterMetadata('StateName','equals','Pennsylvania'); // Pennsylvania Counties

var TheFILTER = ee.Filter.withinDistance( 180000, '.geo', null, '.geo' );

var TheJOIN = ee.Join.simple( );

var JoinedFEATURES = TheJOIN.apply( CountyFEATURES, CityFEATURES, TheFILTER );

print( 'Counties', CountyFEATURES );

print( 'Cities', CityFEATURES );

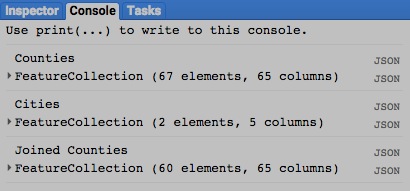
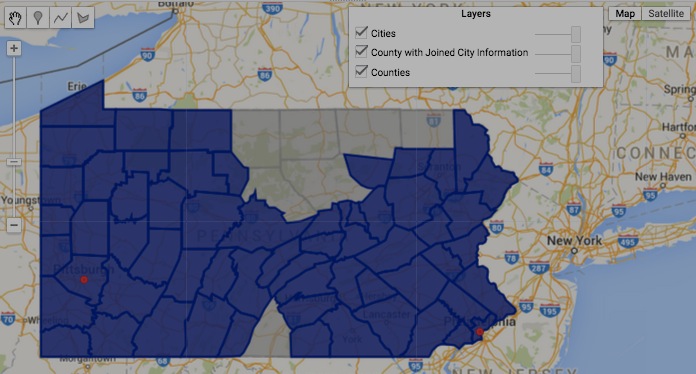
print( 'Joined Counties', JoinedFEATURES );

Map.centerObject( CountyFEATURES, 7 );

Map.addLayer( CountyFEATURES, {color:'999999'}, 'Counties' );

Map.addLayer( JoinedFEATURES, {color:'000088'}, 'County with Joined City Information' );

Map.addLayer( CityFEATURES, {color:'ff0000'}, 'Cities' );



[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** [JOIN](#_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.Join.inverted creates a new join that retains only those elements of a “primary” (image or feature) collection that do not match any element of a “secondary”

(image or feature) collection. Once created, this new join can be applied to specified data by using a **join.apply** operation.

newJoin = ee.Join.inverted( )

The new join

var CityFEATURES = ee.FeatureCollection( 'ft:1G3RZbWoTiCiYv\_LEwc7xKZq8aYoPZlL5\_KuVhyDM' ); // U.S. Cities

var CityFEATURES = CityFEATURES.filterMetadata('city\_name','contains','Pa.'); // Pennsylvania Cities

var CountyFEATURES = ee.FeatureCollection('ft:1S4EB6319wWW2sWQDPhDvmSBIVrD3iEmCLYB7nMM'); // U.S. Counties

var CountyFEATURES = CountyFEATURES.filterMetadata('StateName','equals','Pennsylvania'); // Pennsylvania Counties

var TheJOIN = ee.Join.inverted( );

var TheFILTER = ee.Filter.withinDistance( 180000, '.geo', null, '.geo' );

var JoinedFEATURES = TheJOIN.apply( CountyFEATURES, CityFEATURES, TheFILTER );

print( 'Counties', CountyFEATURES );

print( 'Cities', CityFEATURES );

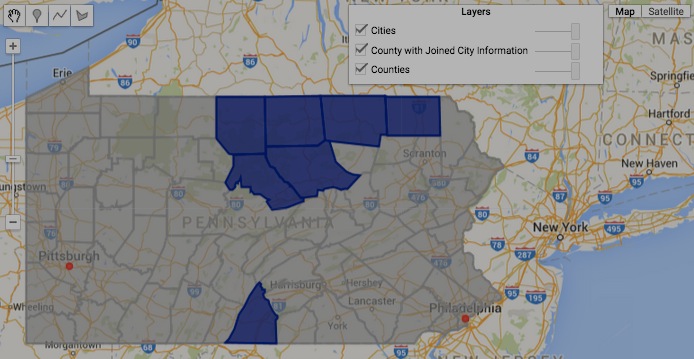
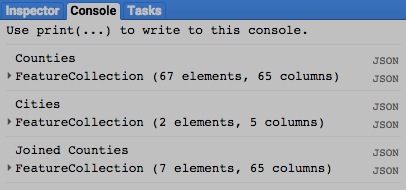
print( 'Joined Counties', JoinedFEATURES );

Map.centerObject( CountyFEATURES, 7 );

Map.addLayer( CountyFEATURES, {color:'999999'}, 'Counties' );

Map.addLayer( JoinedFEATURES, {color:'000088'}, 'County with Joined City Information' );

Map.addLayer( CityFEATURES, {color:'ff0000'}, 'Cities' );



[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** [JOIN](#_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.Join.inner creates a new join that generates a feature collection in which each feature has two properties:

* one representing an element from a specified “primary” collection (of either or both images or features), and
* the other representing a matching element from a specified “secondary collection (of either or both images or features).

If a given primary element matches multiple secondary elements, each match will result in a new feature.

If a given primary element matches no secondary elements, no new feature will be generated.

An additional property can also be optionally included to indicate the measure of the join.

Once created, this new join can be applied to specified data by using a **join.apply** operation.

newJoin = ee.Join.inner ( *primaryPropertyName, secondaryPropertyName, measurePropertyName* )

A specified primary property name,

given as a string. Default: 'primary'

A specified secondary property name,

given as a string. Default: 'secondary'

A specified measure property name,

given as a string. Default: 'null'

The new join

var TheBeatles = ee.FeatureCollection(

[ee.Feature(null, {Item:0, FirstName:'John', SecondName: 'Lennon' }),

ee.Feature(null, {Item:1, FirstName:'Paul', SecondName: 'McCartney'}),

ee.Feature(null, {Item:2, FirstName:'George', SecondName: 'Harrison' }),

ee.Feature(null, {Item:3, FirstName:'Ringo', SecondName: 'Starr' })] );

var ThePresidents = ee.FeatureCollection(

[ee.Feature(null, {Item:0, First:'George', Middle:null, Last:'Washington'} ),

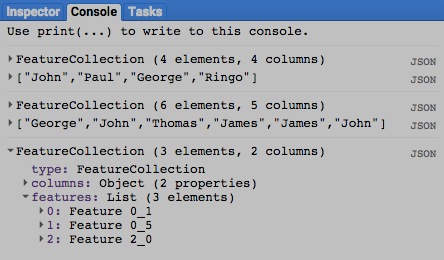
ee.Feature(null, {Item:1, First:'John', Middle:null, Last:'Adams' } ),

ee.Feature(null, {Item:2, First:'Thomas', Middle:null, Last:'Jefferson' } ),

ee.Feature(null, {Item:3, First:'James', Middle:null, Last:'Madison' } ),

ee.Feature(null, {Item:4, First:'James', Middle:null, Last:'Monroe' } ),

ee.Feature(null, {Item:5, First:'John', Middle:'Quincy', Last:'Adams'} )] );

var TheFILTER = ee.Filter.equals( 'FirstName', null, 'First', null );

var TheJOIN = ee.Join.inner( );

var ThePairs = TheJOIN.apply( TheBeatles, ThePresidents, TheFILTER);

print( TheBeatles, TheBeatles.aggregate\_array( 'FirstName' ) );

print( ThePresidents, ThePresidents.aggregate\_array( 'First' ) );

print( ThePairs );

[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** [JOIN](#_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.Join.saveAll creates a new join that retains only those elements of a “primary” (image or feature) collection that match one or more elements of

a “secondary” (image or feature) collection. Each of these retained elements is then augmented with a new property that lists each of its matching elements. An additional property can also be optionally included to indicate a distance measurement (when a **withinDistance**

filter is used by **ee.Join.apply**) or a difference measurement (when a **maxDifference** filter is used). Once created, this new join can be applied to specified data by using a **join.apply** operation.

newJoin = ee.Join.saveAll ( matchPropertyName*, sortProperty, ascendingOrder?, measurePropertyName* )

A string naming the

measurement property.

Default: 'null'

A Boolean set to true (only)

if the list of matching elements

is to be sorted in ascending

order. Default: true

The name to be used for the property that

will list matching elements, given as a string

A string naming a property

by which the list of matching

elements is to be sorted.

Default: null

The new join

var CityFEATURES = ee.FeatureCollection( 'ft:1G3RZbWoTiCiYv\_LEwc7xKZq8aYoPZlL5\_KuVhyDM' ); // U.S. Cities

var CityFEATURES = CityFEATURES.filterMetadata( 'city\_name','contains','Pa.' ); // Pennsylvania Cities

var CountyFEATURES = ee.FeatureCollection( 'ft:1S4EB6319wWW2sWQDPhDvmSBIVrD3iEmCLYB7nMM' ); // U.S. Counties

var CountyFEATURES = CountyFEATURES.filterMetadata( 'StateName','equals','Pennsylvania' ); // Pennsylvania Counties

var TheFILTER = ee.Filter.withinDistance( 100000, '.geo', null, '.geo' );

var TheJOIN = ee.Join.saveAll( 'JOINEDCITIES', null, true,'DISTANCE' );

var JoinedFEATURES = TheJOIN.apply( CountyFEATURES, CityFEATURES, TheFILTER );

print( 'Counties', CountyFEATURES );

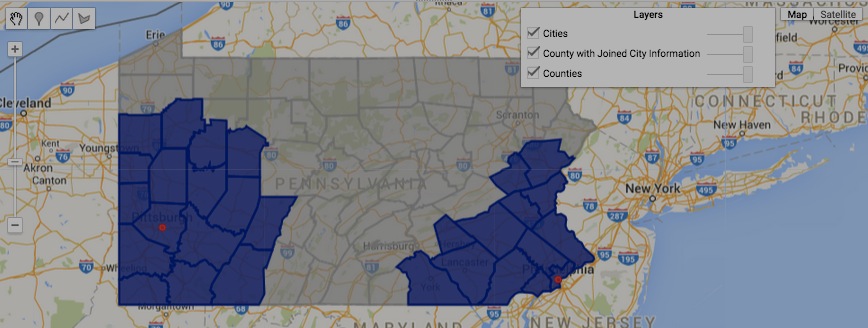
print( 'Cities', CityFEATURES );

print( 'Joined Counties', JoinedFEATURES );

Map.centerObject( CountyFEATURES, 7 );

Map.addLayer( CountyFEATURES, {color:'999999'}, 'Counties' );

Map.addLayer( JoinedFEATURES, {color:'000088'}, 'County with Joined City Information' );

Map.addLayer( CityFEATURES, {color:'ff0000'}, 'Cities' );

STILL UNABLE TO ACCESS THE RESULTING ‘DISTANCE’ PROPERTY

[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** [JOIN](#_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.Join.saveBest creates a new join that retains only those elements of a “primary” (image or feature) collection that match one or more elements of

a “secondary” (image or feature) collection. Each of these retained elements is then augmented with a new property that contains

whichever one of its matching elements yields the lowest distance measurement (when a **withinDistance** filter is used by **ee.Join.apply**)

or a difference measurement (when a **maxDifference** filter is used). An additional property can also be optionally included to indicate

the value of that measurement. Once created, this new join can be applied to specified data by using a **join.apply** operation.

newJoin = ee.Join.saveBest ( matchPropertyName, measurePropertyName )

The name of the measurement property, given as a string

The name to be used for the property that will list matching elements, given as a string

The new join

var CityFEATURES = ee.FeatureCollection( 'ft:1G3RZbWoTiCiYv\_LEwc7xKZq8aYoPZlL5\_KuVhyDM' ); // U.S. Cities

var CityFEATURES = CityFEATURES.filterMetadata('city\_name','contains','Pa.'); // Pennsylvania Cities

var CountyFEATURES = ee.FeatureCollection('ft:1S4EB6319wWW2sWQDPhDvmSBIVrD3iEmCLYB7nMM'); // U.S. Counties

var CountyFEATURES = CountyFEATURES.filterMetadata('StateName','equals','Pennsylvania'); // Pennsylvania Counties

var TheFILTER = ee.Filter.withinDistance( 100000, '.geo', null, '.geo' );

var TheJOIN = ee.Join.saveBest('JOINEDCITY', 'DISTANCEFROMCITY');

var JoinedFEATURES = TheJOIN.apply( CountyFEATURES, CityFEATURES, TheFILTER );

print( 'Counties', CountyFEATURES );

print( 'Cities', CityFEATURES );

print( 'Joined Counties', JoinedFEATURES );

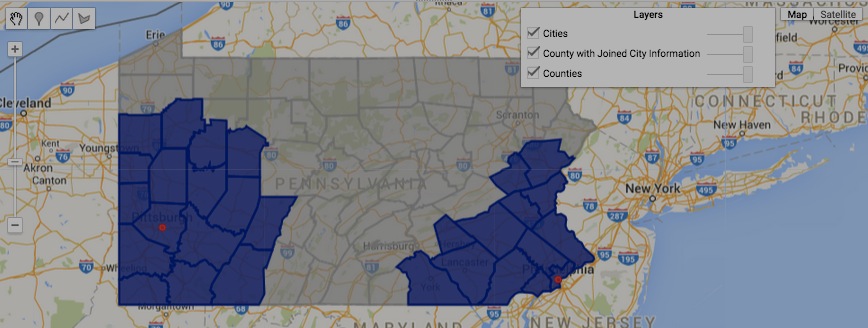
Map.centerObject( CountyFEATURES, 7 );

Map.addLayer( CountyFEATURES, {color:'999999'}, 'Counties' );

Map.addLayer( JoinedFEATURES, {color:'000088'}, 'County with Joined City Information' );

Map.addLayer( CityFEATURES, {color:'ff0000'}, 'Cities' );

// How to display just those counties that had Pittsburgh joined to them



[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** [JOIN](#_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.Join.saveFirst creates a new join that retains only those elements of a “primary” (image or feature) collection that match one or more elements of

a “secondary” (image or feature) collection. Each of these retained elements is then augmented with a new property that contains

whichever one of its matching elements occurs first when those matching elements are sorted on a specified property in a specified order.

An additional property can also be optionally included to indicate a distance measurement (when a **withinDistance** filter is used by **ee.Join.apply**) or a difference measurement (when a **maxDifference** filter is used). Once created, this new join can be applied to specified

data by using a **join.apply** operation.

newJoin = ee.Join.saveFirst( matchPropertyName*, sortProperty, ascendingOrder?, measurePropertyName* )

A string naming the

measurement property.

Default: 'null'

A Boolean set to true (only)

if the list of matching elements

is to be sorted in ascending

order. Default: true

The name to be used for the property that

will list matching elements, given as a string

A string naming a property

by which the list of matching

elements is to be sorted.

Default: null

The new join

var CityFEATURES = ee.FeatureCollection( 'ft:1G3RZbWoTiCiYv\_LEwc7xKZq8aYoPZlL5\_KuVhyDM' ); // U.S. Cities

var CityFEATURES = CityFEATURES.filterMetadata('city\_name','contains','Pa.'); // Pennsylvania Cities

var CountyFEATURES = ee.FeatureCollection('ft:1S4EB6319wWW2sWQDPhDvmSBIVrD3iEmCLYB7nMM'); // U.S. Counties

var CountyFEATURES = CountyFEATURES.filterMetadata('StateName','equals','Pennsylvania'); // Pennsylvania Counties

var TheFILTER = ee.Filter.withinDistance( 100000, '.geo', null, '.geo' );

var TheJOIN = ee.Join.saveFirst('JOINEDCITY', 'DISTANCEFROMCITY');

var JoinedFEATURES = TheJOIN.apply( CountyFEATURES, CityFEATURES, TheFOLTER );

print( 'Counties', CountyFEATURES );

print( 'Cities', CityFEATURES );

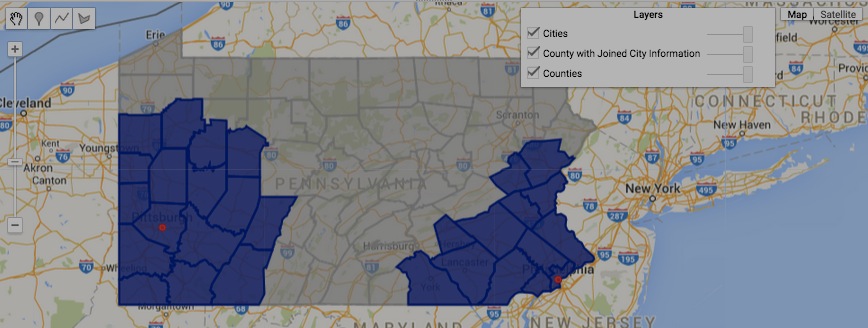
print( 'Joined Counties', JoinedFEATURES );

Map.centerObject( CountyFEATURES, 7 );

Map.addLayer( CountyFEATURES, {color:'999999'}, 'Counties' );

Map.addLayer( JoinedFEATURES, {color:'000088'}, 'County with Joined City Information' );

Map.addLayer( CityFEATURES, {color:'ff0000'}, 'Cities' );

// How to display just those counties

that had Pittsburgh joined to them

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

join.apply creates a new image or feature collection by subjecting two specified image or feature collections to a specified join.

newCollection = ee.apply( primaryCollection, secondaryCollection, filter )

The new image or

feature collection

The first of the two

specified collections

The first of the two

specified collections

The specified

filter

var TheBeatles = ee.FeatureCollection(

[ee.Feature(null, {Item:0, FirstName:'John', SecondName: 'Lennon' }),

ee.Feature(null, {Item:1, FirstName:'Paul', SecondName: 'McCartney'}),

ee.Feature(null, {Item:2, FirstName:'George', SecondName: 'Harrison' }),

ee.Feature(null, {Item:3, FirstName:'Ringo', SecondName: 'Starr' })] );

var ThePresidents = ee.FeatureCollection(

[ee.Feature(null, {Item:0, First:'George', Middle:null, Last:'Washington'} ),

ee.Feature(null, {Item:1, First:'John', Middle:null, Last:'Adams' } ),

ee.Feature(null, {Item:2, First:'Thomas', Middle:null, Last:'Jefferson' } ),

ee.Feature(null, {Item:3, First:'James', Middle:null, Last:'Madison' } ),

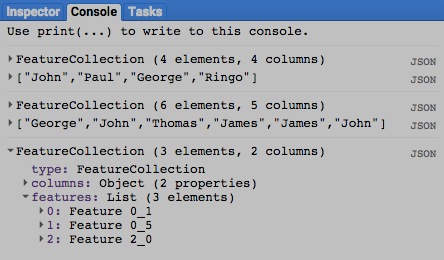
ee.Feature(null, {Item:4, First:'James', Middle:null, Last:'Monroe' } ),

ee.Feature(null, {Item:5, First:'John', Middle:'Quincy', Last:'Adams'} )] );

var TheFILTER = ee.Filter.equals( 'FirstName', null, 'First', null );

var TheJOIN = ee.Join.inner( );

var ThePairs = TheJOIN.apply( TheBeatles, ThePresidents, TheFILTER);

print( TheBeatles, TheBeatles.aggregate\_array( 'FirstName' ) );

print( ThePresidents, ThePresidents.aggregate\_array( 'First' ) );

print( ThePairs );

[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** [JOIN](#_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 join.getInfo

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

newObject = ee.Algorithms.Describe( oldJoin )

and oldJoin.getInfo( )

The specified join

The new object

var TheJOIN = ee.Join.simple( );

print( 'From print:', TheJOIN );

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

print( 'From getInfo( ):', TheJOIN.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** [JOIN](#_top) [[PARAMETERS](EE26%20%20%20%20%20%20%20%20%20Parameters.docx)](EE25%20%20%20%20%20%20%20%20%20Parameters.docx)

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

newString = oldJoin.toString ( )

and oldJoin.serialize( )

The specified join

The new string

var TheJOIN = ee.Join.simple( );

print( 'From print:', TheJOIN );

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

print( 'From serialize( ):', TheJOIN.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** [JOIN](#_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 ( join ) and console.log ( join ) present JSON-formatted text renditions of a specified join in the console.

print( oldJoin ) or console.log( oldJoin )

The specified join

var TheJOIN = ee.Join.simple( );

print( 'From print:', TheJOIN );

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

[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** [JOIN](#_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 ( join ) and confirm ( join ) present JSON-formatted text renditions of a specified

date in a pop-up message box.

alert( oldJoin ) or confirm( oldJoin )

The specified join

var TheJOIN = ee.Join.simple( );

alert( TheJOIN );

confirm( TheJOIN );