Single-Page API Reference

ee.Algorithms.CannyEdgeDetector

Applies the Canny edge detection algorithm to an image. The output is an image whose bands have the same names as the input bands, and in which non-zero values indicate edges, and the magnitude of the value is the gradient magnitude.

Usage	Returns
`ee.Algorithms.CannyEdgeDetector(image, threshold, sigma)`	Image

Argument	Type	Details
`image`	Image	The image on which to apply edge detection.
`threshold`	Float	Threshold value. The pixel is only considered for edge detection if the gradient magnitude is higher than this threshold.
`sigma`	*Float, default: 1*	*Sigma value for a gaussian filter applied before edge detection. 0 means apply no filtering.*

ee.Algorithms.Collection

Returns a Collection containing the specified features.

Usage	Returns
`ee.Algorithms.Collection(features)`	FeatureCollection

Argument	Type	Details
`features`	List	The features comprising the collection.

ee.Algorithms.CrossCorrelation

Gives information on the quality of image registration between two (theoretically) co-registered images. The input is two images with the same number of bands. This function outputs an image composed of four bands of information. The first three are distances: the deltaX, deltaY, and the Euclidean distance for each pixel in imageA to the pixel which has the highest corresponding correlation coefficient in imageB. The fourth band is the value of the correlation coefficient for that pixel [-1 : +1].

Usage	Returns
`ee.Algorithms.CrossCorrelation(imageA, imageB, maxGap, windowSize, maxMaskedFrac)`	Image

Argument	Type	Details
`imageA`	Image	First image, with N bands.
`imageB`	Image	Second image, must have the same number of bands as imageA.
`maxGap`	Integer	The greatest distance a pixel may shift in either X or Y.
`windowSize`	Integer	Size of the window to be compared.
`maxMaskedFrac`	*Float, default: 0*	*The maximum fraction of pixels within the correlation window that are allowed to be masked. This test is applied at each offset location within the search region. For each offset, the overlapping image patches are compared and a correlation score computed. A pixel within these overlapping patches is considered masked if either of the patches is masked there. If the test fails at any single location in the search region, the output pixel for which the correlation is being computed is considered invalid, and will be masked.*

ee.Algorithms.Date

Creates a Date.

Usage	Returns
`ee.Algorithms.Date(value, timeZone)`	Date

Argument	Type	Details
`value`	Object	A number (interpreted as milliseconds since 1970-01-01T00:00:00Z), or string such as '1996-01-01' or '1996-001' or '1996-01-01T08:00'.
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Algorithms.Describe

Describes an object using a simple JSON-compatible structure.

Usage	Returns
`ee.Algorithms.Describe(input)`	Object

Argument	Type	Details
`input`	Object	The object to describe.

ee.Algorithms.Dictionary

Constructs a dictionary.

Usage	Returns
`ee.Algorithms.Dictionary(`*`input`*`)`	Dictionary

Argument	Type	Details
`input`	*Object, default: null*	*An object to convert to a dictionary. Either a JSON dictionary or a list of alternating key/value pairs. Keys must be strings.*

ee.Algorithms.FMask.fillMinima

Fills local minima. Only works on INT types.

Usage	Returns
`ee.Algorithms.FMask.fillMinima(image, `*`borderValue`*`, `*`neighborhood`*`)`	Image

Argument	Type	Details
`image`	Image	The image to fill.
`borderValue`	*Long, default: null*	*The border value.*
`neighborhood`	*Integer, default: 50*	*The size of the neighborhood to compute over.*

ee.Algorithms.FMask.matchClouds

Runs the FMask cloud and shadow matching. Outputs a single band ('csm'), containing the computed cloud and shadow masks.

Usage	Returns
`ee.Algorithms.FMask.matchClouds(input, cloud, shadow, btemp, sceneLow, sceneHigh, neighborhood)`	Image

Argument	Type	Details
`input`	Image	The scene for which to compute cloud and shadow masks.
`cloud`	Image	Potential cloud mask image. Expected to contain 1s for cloudy pixels and masked pixels everywhere else.
`shadow`	Image	Potential shadow mask image. Expected to contain 1s for shadow pixels and masked pixels everywhere else.
`btemp`	Image	Brightness temperature image, in Celsius.
`sceneLow`	Float	The 0.175 percentile brightness temperature of the scene.
`sceneHigh`	Float	The 0.825 percentile brightness temperature of the scene.
`neighborhood`	*Integer, default: 50*	*The neighborhood to pad around each tile.*

ee.Algorithms.Feature

Returns a Feature composed of the given geometry and metadata.

Usage	Returns
`ee.Algorithms.Feature(geometry, metadata, geometryKey)`	Feature

Argument	Type	Details
`geometry`	*Geometry, default: null*	*The geometry of the feature.*
`metadata`	*Dictionary, default: {}*	*The properties of the feature.*
`geometryKey`	*String, default: null*	*Obsolete; has no effect.*

ee.Algorithms.GeometryConstructors.BBox

Constructs a rectangle whose edges are lines of latitude and longitude.

The result is a planar WGS84 rectangle.

If (east - west) ≥ 360 then the longitude range will be normalized to -180 to +180; otherwise they will be treated as designating points on a circle (e.g. east may be numerically less than west).

Usage	Returns
`ee.Algorithms.GeometryConstructors.BBox(west, south, east, north)`	Geometry

Argument	Type	Details
`west`	Float	The westernmost enclosed longitude. Will be adjusted to lie in the range -180 to 180.
`south`	Float	The southernmost enclosed latitude. If less than -90 (south pole), will be treated as -90.
`east`	Float	The easternmost enclosed longitude.
`north`	Float	The northernmost enclosed latitude. If greater than +90 (north pole), will be treated as +90.

ee.Algorithms.GeometryConstructors.LineString

Constructs a LineString from the given coordinates.

Usage	Returns
`ee.Algorithms.GeometryConstructors.LineString(coordinates, crs, geodesic)`	Geometry

Argument	Type	Details
`coordinates`	List	The list of Points or pairs of Numbers in x,y order.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, default: null*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*

ee.Algorithms.GeometryConstructors.LinearRing

Constructs a LinearRing from the given coordinates, automatically adding the first point at the end if the ring is not explicitly closed.

Usage	Returns
`ee.Algorithms.GeometryConstructors.LinearRing(coordinates, crs, geodesic)`	Geometry

Argument	Type	Details
`coordinates`	List	The list of Points or pairs of Numbers in x,y order.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, default: null*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*

ee.Algorithms.GeometryConstructors.MultiGeometry

Constructs a MultiGeometry from the given list of geometry elements.

Usage	Returns
`ee.Algorithms.GeometryConstructors.MultiGeometry(geometries, crs, geodesic, maxError)`	Geometry

Argument	Type	Details
`geometries`	List	The list of geometries for the MultiGeometry.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, default: null*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, default: null*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*

ee.Algorithms.GeometryConstructors.MultiLineString

Constructs a MultiLineString from the given coordinates.

Usage	Returns
`ee.Algorithms.GeometryConstructors.MultiLineString(coordinates, crs, geodesic, maxError)`	Geometry

Argument	Type	Details
`coordinates`	List	The list of LineStrings, or to wrap a single LineString, the list of Points or pairs of Numbers in x,y order.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, default: null*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, default: null*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*

ee.Algorithms.GeometryConstructors.MultiPoint

Constructs a MultiPoint from the given coordinates.

Usage	Returns
`ee.Algorithms.GeometryConstructors.MultiPoint(coordinates, crs)`	Geometry

Argument	Type	Details
`coordinates`	List	The list of Points or pairs of Numbers in x,y order.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*

ee.Algorithms.GeometryConstructors.MultiPolygon

Constructs a MultiPolygon from the given coordinates.

Usage	Returns
`ee.Algorithms.GeometryConstructors.MultiPolygon(coordinates, crs, geodesic, maxError, evenOdd)`	Geometry

Argument	Type	Details
`coordinates`	List	A list of Polygons, or for one simple polygon, a list of Points or pairs of Numbers in x,y order.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, default: null*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, default: null*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*
`evenOdd`	*Boolean, default: true*	*If true, polygon interiors will be determined by the even/odd rule, where a point is inside if it crosses an odd number of edges to reach a point at infinity. Otherwise polygons use the left-inside rule, where interiors are on the left side of the shell's edges when walking the vertices in the given order.*

ee.Algorithms.GeometryConstructors.Point

Constructs a new Point from the given x,y coordinates.

Usage	Returns
`ee.Algorithms.GeometryConstructors.Point(coordinates, crs)`	Geometry

Argument	Type	Details
`coordinates`	List	The coordinates of this Point in x,y order.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*

ee.Algorithms.GeometryConstructors.Polygon

Constructs a Polygon from the given coordinates.

Usage	Returns
`ee.Algorithms.GeometryConstructors.Polygon(coordinates, crs, geodesic, maxError, evenOdd)`	Geometry

Argument	Type	Details
`coordinates`	List	A list of LinearRings where the first is the shell and the rest are holes, or for a simple polygon, a list of Points or pairs of Numbers in x,y order.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, default: null*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, default: null*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*
`evenOdd`	*Boolean, default: true*	*If true, polygon interiors will be determined by the even/odd rule, where a point is inside if it crosses an odd number of edges to reach a point at infinity. Otherwise polygons use the left-inside rule, where interiors are on the left side of the shell's edges when walking the vertices in the given order.*

ee.Algorithms.GeometryConstructors.Rectangle

Constructs a rectangular polygon from the given corner points.

Usage	Returns
`ee.Algorithms.GeometryConstructors.Rectangle(coordinates, crs, geodesic, evenOdd)`	Geometry

Argument	Type	Details
`coordinates`	List	The low and then high corners of the Rectangle, as a list of Points or pairs of Numbers in x,y order.
`crs`	*Projection, default: null*	*The coordinate reference system of the coordinates. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, default: null*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`evenOdd`	*Boolean, default: true*	*If true, polygon interiors will be determined by the even/odd rule, where a point is inside if it crosses an odd number of edges to reach a point at infinity. Otherwise polygons use the left-inside rule, where interiors are on the left side of the shell's edges when walking the vertices in the given order.*

ee.Algorithms.HillShadow

Creates a shadow band, with output 1 where pixels are illumunated and 0 where they are shadowed. Takes as input an elevation band, azimuth and zenith of the light source in degrees, a neighborhood size, and whether or not to apply hysteresis when a shadow appears. Currently, this algorithm only works for Mercator projections, in which light rays are parallel.

Usage	Returns
`ee.Algorithms.HillShadow(image, azimuth, zenith, neighborhoodSize, hysteresis)`	Image

Argument	Type	Details
`image`	Image	The image to which to apply the shadow algorithm, in which each pixel should represent an elevation in meters.
`azimuth`	Float	Azimuth in degrees.
`zenith`	Float	Zenith in degrees.
`neighborhoodSize`	*Integer, default: 0*	*Neighborhood size.*
`hysteresis`	*Boolean, default: false*	*Use hysteresis. Less physically accurate, but may generate better images.*

ee.Algorithms.HoughTransform

Applies the Hough transform to an image. For every input band, outputs a band where lines are detected by thresholding the Hough transform with a value of lineThreshold. The output band is named [input]_lines, where [input] is the name of the original band. The defaults provided for the parameters are intended as a starting point for use with UINT8 images.

Usage	Returns
`ee.Algorithms.HoughTransform(image, gridSize, inputThreshold, lineThreshold, smooth)`	Image

Argument	Type	Details
`image`	Image	The image to which to apply the transform.
`gridSize`	*Integer, default: 256*	*The size of the grid over which to perform the computation.*
`inputThreshold`	*Float, default: 64*	*Value threshold for input image. Pixels equal to or above this value are considered active.*
`lineThreshold`	*Float, default: 72*	*Threshold for line detection. Values equal to or above this threshold on the Hough transform are considered to be detected lines.*

Argument	Type	Details
`smooth`	*Boolean, default: true*	*Whether to smooth the Hough transform before line detection.*

ee.Algorithms.If

Selects one of its inputs based on a condition, similar to an if-then-else construct.

Usage	Returns
`ee.Algorithms.If(`*`condition`*`, `*`trueCase`*`, `*`falseCase`*`)`	Object

Argument	Type	Details
`condition`	*Object, default: null*	*The condition that determines which result is returned. If this is not a boolean, it is interpreted as a boolean by the following rules:* • *Numbers that are equal to 0 or a NaN are false.* • *Empty strings, lists and dictionaries are false.* • *Null is false.* • *Everything else is true.*
`trueCase`	*Object, default: null*	*The result to return if the condition is true.*
`falseCase`	*Object, default: null*	*The result to return if the condition is false.*

ee.Algorithms.Image.Segmentation.GMeans

Performs G-Means clustering on the input image. Iteratively applies k-means followed by a normality test to automatically determine the number of clusters to use. The output contains a 'clusters' band containing the integer ID of the cluster that each pixel belongs to. The algorithm can work either on a fixed grid of non-overlapping cells (gridSize, which can be smaller than a tile) or on tiles with overlap (neighborhoodSize). The default is to use tiles with no overlap. Clusters in one cell or tile are unrelated to clusters in another. Any cluster that spans a cell or tile boundary may receive two different labels in the two halves. Any input pixels with partial masks are fully masked in the output. This algorithm is only expected to perform well for images with a narrow dynamic range (i.e. bytes or shorts).

See: G. Hamerly and C. Elkan. 'Learning the k in k-means'. NIPS, 2003.

Usage	Returns
`ee.Algorithms.Image.Segmentation.GMeans(image, numIterations, pValue, neighborhoodSize, gridSize, uniqueLabels)`	Image

Argument	Type	Details
`image`	Image	The input image for clustering.
`numIterations`	*Integer, default: 10*	*Number of iterations. Default 10.*
`pValue`	*Float, default: 50*	*Significance level for normality test.*
`neighborhoodSize`	*Integer, default: 0*	*Neighborhood size. The amount to extend each tile (overlap) when computing the clusters. This option is mutually exclusive with gridSize.*
`gridSize`	*Integer, default: null*	*Grid cell-size. If greater than 0, kMeans will be run independently on cells of this size. This has the effect of limiting the size of any cluster to be gridSize or smaller. This option is mutually exclusive with neighborhoodSize.*
`uniqueLabels`	*Boolean, default: true*	*If true, clusters are assigned unique IDs. Otherwise, they repeat per tile or grid cell.*

ee.Algorithms.Image.Segmentation.KMeans

Performs K-Means clustering on the input image. Outputs a 1-band image containing the ID of the cluster that each pixel belongs to. The algorithm can work either on a fixed grid of non-overlapping cells (gridSize, which can be smaller than a tile) or on tiles with overlap (neighborhoodSize). The default is to use tiles with no overlap. Clusters in one cell or tile are unrelated to clusters in another. Any cluster that spans a cell or tile boundary may receive two different labels in the two halves. Any input pixels with partial masks are fully masked in the output.

Usage	Returns
`ee.Algorithms.Image.Segmentation.KMeans(image, numClusters, numIterations, neighborhoodSize, gridSize, forceConvergence, uniqueLabels)`	Image

Argument	Type	Details
`image`	Image	The input image for clustering.
`numClusters`	*Integer, default: 8*	*Number of clusters.*

Argument	Type	Details
`numIterations`	*Integer, default: 20*	*Number of iterations.*
`neighborhoodSize`	*Integer, default: 0*	*Neighborhood size. The amount to extend each tile (overlap) when computing the clusters. This option is mutually exclusive with gridSize.*
`gridSize`	*Integer, default: null*	*Grid cell-size. If greater than 0, kMeans will be run independently on cells of this size. This has the effect of limiting the size of any cluster to be gridSize or smaller. This option is mutually exclusive with neighborhoodSize.*
`forceConvergence`	*Boolean, default: false*	*If true, an error is thrown if convergence is not achieved before numIterations.*
`uniqueLabels`	*Boolean, default: true*	*If true, clusters are assigned unique IDs. Otherwise, they repeat per tile or grid cell.*

ee.Algorithms.Image.Segmentation.SNIC

Superpixel clustering based on SNIC (Simple Non-Iterative Clustering). Outputs a band of cluster IDs and the per-cluster averages for each of the input bands. If the 'seeds' image isn't provided as input, the output will include a 'seeds' band containing the generated seed locations. See: Achanta, Radhakrishna and Susstrunk, Sabine, 'Superpixels and Polygons using Simple Non-Iterative Clustering', CVPR, 2017.

Usage	Returns
`ee.Algorithms.Image.Segmentation.SNIC(image, size, compactness, connectivity, neighborhoodSize, seeds)`	Image

Argument	Type	Details
`image`	Image	The input image for clustering.
`size`	*Integer, default: 5*	*The superpixel seed location spacing, in pixels. If 'seeds' image is provided, no grid is produced.*
`compactness`	*Float, default: 1*	*Compactness factor. Larger values cause clusters to be more compact (square). Setting this to 0 disables spatial distance weighting.*
`connectivity`	*Integer, default: 8*	*Connectivity. Either 4 or 8.*
`neighborhoodSize`	*Integer, default: null*	*Tile neighborhood size (to avoid tile boundary artifacts). Defaults to 2 * size.*
`seeds`	*Image, default: null*	*If provided, any non-zero valued pixels are used as seed locations. Pixels that touch (as specified by 'connectivity') are considered to belong to the same cluster.*

ee.Algorithms.Image.Segmentation.seedGrid

Selects seed pixels for clustering.

Usage	Returns
`ee.Algorithms.Image.Segmentation.seedGrid(`*`size`*`, `*`gridType`*`)`	Image

Argument	Type	Details
`size`	*Integer, default: 5*	*The superpixel seed location spacing, in pixels.*
`gridType`	*String, default: "square"*	*Type of grid. One of 'square' or 'hex'.*

ee.Algorithms.IsEqual

Returns whether two objects are equal.

Usage	Returns
`ee.Algorithms.IsEqual(`*`left`*`, `*`right`*`)`	Boolean

Argument	Type	Details
`left`	*Object, default: null*	
`right`	*Object, default: null*	

ee.Algorithms.Landsat.TOA

Calibrates Landsat DN to TOA reflectance and brightness temperature for Landsat and similar data. For recently-acquired scenes calibration coefficients are extracted from the image metadata; for older scenes the coefficients are derived from:

Chander, Gyanesh, Brian L. Markham, and Dennis L. Helder. "Summary of current radiometric calibration coefficients for Landsat MSS, TM, ETM+, and EO-1 ALI sensors." Remote sensing of environment 113.5 (2009): 893-903.

Usage	Returns
`ee.Algorithms.Landsat.TOA(input)`	Image

Argument	Type	Details
`input`	Image	The Landsat image to process.

ee.Algorithms.Landsat.calibratedRadiance

Calibrates each band of an image by applying linear transformation with slope `RADIANCE_MULT_BAND_N` and y-intercept `RADIANCE_ADD_BAND_N`; these values are extracted from the image metadata.

Usage	Returns
`ee.Algorithms.Landsat.calibratedRadiance(image)`	Image

Argument	Type	Details
`image`	Image	The input Landsat image.

ee.Algorithms.Landsat.pathRowLimit

Limits requests to an ImageCollection of Landsat scenes to return a controllable number of the best scenes for each request. This is intended for use with statistical algorithms like median composites that need a certain amount of good data to perform well, but that do not benefit substantially from additional data beyond that while becoming needlessly expensive. The default arguments select approximately one year's worth of good data.

Note that in rare circumstances, when the tile boundary aligns with a Landsat WRS cell boundary, queries for adjacent tiles may yield conflicting results. This is why it is important that this algorithm only be used with statistical methods that can tolerate these inconsistencies.

Usage	Returns
`ee.Algorithms.Landsat.pathRowLimit(collection, `*`maxScenesPerPathRow`*`, `*`maxScenesTotal`*`)`	ImageCollection

Argument	Type	Details
`collection`	ImageCollection	The Landsat ImageCollection to limit.
`maxScenesPerPathRow`	*Integer, default: 25*	*The max number of scenes to return per path/row.*
`maxScenesTotal`	*Integer, default: 100*	*The max number of scenes to return per request total.*

ee.Algorithms.Landsat.simpleCloudScore

Computes a simple cloud-likelihood score in the range [0,100] using a combination of brightness, temperature, and NDSI. This is not a robust cloud detector, and is intended mainly to compare multiple looks at the same point for *relative* cloud likelihood.

Usage	Returns
`ee.Algorithms.Landsat.simpleCloudScore(image)`	Image

Argument	Type	Details
`image`	Image	The Landsat TOA image to process.

ee.Algorithms.Landsat.simpleComposite

Computes a Landsat TOA composite from a collection of raw Landsat scenes. It applies standard TOA calibration and then assigns a cloud score to each pixel using the SimpleLandsatCloudScore algorithm. It selects the lowest possible range of cloud scores at each point and then computes per-band percentile values from the accepted pixels. This algorithm also uses the LandsatPathRowLimit algorithm to select only the least-cloudy scenes in regions where more than maxDepth input scenes are available.

Usage	Returns
`ee.Algorithms.Landsat.simpleComposite(collection, `*`percentile`*`, `*`cloudScoreRange`*`, `*`maxDepth`*`, `*`asFloat`*`)`	Image

Argument	Type	Details
`collection`	ImageCollection	The raw Landsat ImageCollection to composite.
`percentile`	*Integer, default: 50*	*The percentile value to use when compositing each band.*
`cloudScoreRange`	*Integer, default: 10*	*The size of the range of cloud scores to accept per pixel.*
`maxDepth`	*Integer, default: 40*	*An approximate limit on the maximum number of scenes used to compute each pixel.*
`asFloat`	*Boolean, default: false*	*If true, output bands are in the same units as the Landsat.TOA algorithm; if false, TOA values are converted to uint8 by multiplying by 255 (reflective bands) or subtracting 100 (thermal bands) and rounding to the nearest integer.*

ee.Algorithms.ObjectType

Returns a string representing the type of the given object.

Usage	Returns
`ee.Algorithms.ObjectType(`*`value`*`)`	String

Argument	Type	Details
`value`	*Object, default: null*	*The object to get the type of.*

ee.Algorithms.ProjectionTransform

Transforms the geometry of a feature to a specific projection.

Usage	Returns
`ee.Algorithms.ProjectionTransform(feature, `*`proj`*`, `*`maxError`*`)`	Feature

Argument	Type	Details
`feature`	Element	The feature the geometry of which is being converted.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Algorithms.Sentinel2.CDI

Computes the Cloud Displacement Index (CDI) from a Sentinel-2 Level 1C image. CDI is a measure of the optical separation in elevated objects due to sensor parallax. Returns a floating point band named "cdi".

See Frantz, D., Hass, E., Uhl, A., Stoffels, J., & Hill, J. (2018). Improvement of the Fmask algorithm for Sentinel-2 images: Separating clouds from bright surfaces based on parallax effects. Remote sensing of environment, 215, 471-481.

Usage	Returns
`ee.Algorithms.Sentinel2.CDI(source)`	Image

Argument	Type	Details
`source`	Image	The source image.

ee.Algorithms.String

Converts the input to a string.

Usage	Returns
`ee.Algorithms.String(input)`	String

Argument	Type	Details
`input`	Object	The object to convert.

ee.Algorithms.TemporalSegmentation.Ccdc

Implements the Continuous Change Detection and Classification temporal breakpoint algorithm. This algorithm finds temporal breakpoints in an image collection by iteratively fitting harmonic functions to the data. Fit coefficients are produced for all input bands, but the bands used for breakpoint detection can be specified with the 'breakpointBands' argument.

For more details, see Zhu, Z. and Woodcock, C.E., 2014. Continuous change detection and classification of land cover using all available Landsat data. Remote sensing of Environment, 144, pp.152-171.

Usage	Returns
`ee.Algorithms.TemporalSegmentation.Ccdc(collection, `*`breakpointBands`*`, `*`tmaskBands`*`, `*`minObservations`*`, `*`chiSquareProbability`*`, `*`minNumOfYearsScaler`*`, `*`dateFormat`*`, `*`lambda`*`, `*`maxIterations`*`)`	Image

Argument	Type	Details
`collection`	ImageCollection	Collection of images on which to run CCDC.
`breakpointBands`	*List, default: null*	*The name or index of the bands to use for change detection. If unspecified, all bands are used.*
`tmaskBands`	*List, default: null*	*The name or index of the bands to use for iterative TMask cloud detection. These are typically the green band and the SWIR1 band. If unspecified, TMask is not used. If specified, 'tmaskBands' must be included in 'breakpointBands'.*

Argument	Type	Details
`minObservations`	*Integer, default: 6*	*The number of observations required to flag a change.*
`chiSquareProbability`	*Float, default: 0.99*	*The chi-square probability threshold for change detection in the range of [0, 1]*
`minNumOfYearsScaler`	*Float, default: 1.33*	*Factors of minimum number of years to apply new fitting.*
`dateFormat`	*Integer, default: 0*	*The time representation to use during fitting: 0 = jDays, 1 = fractional years, 2 = unix time in milliseconds. The start, end and break times for each temporal segment will be encoded this way.*
`lambda`	*Float, default: 20*	*Lambda for LASSO regression fitting. If set to 0, regular OLS is used instead of LASSO.*
`maxIterations`	*Integer, default: 25000*	*Maximum number of runs for LASSO regression convergence. If set to 0, regular OLS is used instead of LASSO.*

ee.Algorithms.TemporalSegmentation.Ewmacd

Exponentially Weighted Moving Average Change Detection. This algorithm computes a harmonic model for the 'training' portion of the input data and subtracts that from the original results. The residuals are then subjected to Shewhart X-bar charts and an exponentially weighted moving average. Disturbed pixels are indicated when the charts signal a deviation from the given control limits.

The output is a 5 band image containing the bands:

ewma: a 1D array of the EWMA score for each input image. Negative values represent disturbance and positive values represent recovery.

harmonicCoefficients: A 1-D array of the computed harmonic coefficient pairs. The coefficients are ordered as [constant, sin0, cos0, sin1, cos1...]

rmse: the RMSE from the harmonic regression.

rSquared: r-squared value from the harmonic regression.

residuals: 1D array of residuals from the harmonic regression.

See: Brooks, E.B., Wynne, R.H., Thomas, V.A., Blinn, C.E. and Coulston, J.W., 2014. On-the-fly massively multitemporal change detection using statistical quality control charts and Landsat data. IEEE Transactions on Geoscience and Remote Sensing, 52(6), pp.3316-3332.

Usage	Returns
`ee.Algorithms.TemporalSegmentation.Ewmacd(timeSeries, vegetationThreshold, trainingStartYear, trainingEndYear, harmonicCount, xBarLimit1, xBarLimit2, lambda, lambdasigs, rounding, persistence)`	Image

Argument	Type	Details
`timeSeries`	ImageCollection	Collection from which to extract EWMA. This collection is expected to contain 1 image for each year and be sorted temporally.
`vegetationThreshold`	Float	Threshold for vegetation. Values below this are considered non-vegetation.
`trainingStartYear`	Integer	Start year of training period, inclusive.
`trainingEndYear`	Integer	End year of training period, exclusive.
`harmonicCount`	*Integer, default: 2*	*Number of harmonic function pairs (sine and cosine) used.*
`xBarLimit1`	*Float, default: 1.5*	*Threshold for initial training xBar limit.*
`xBarLimit2`	*Integer, default: 20*	*Threshold for running xBar limit.*
`lambda`	*Float, default: 0.3*	*The 'lambda' tuning parameter weighting new years vs the running average.*
`lambdasigs`	*Float, default: 3*	*EWMA control bounds, in units of standard deviations.*
`rounding`	*Boolean, default: true*	*Should rounding be performed for EWMA*
`persistence`	*Integer, default: 3*	*Minimum number of observations needed to consider a change.*

ee.Algorithms.TemporalSegmentation.LandTrendr

Landsat-based detection of Trends in Disturbance and Recovery: temporally segments a time-series of images by extracting the spectral trajectories of change over time. The first band of each image is used to find breakpoints, and those breakpoints are used to perform fitting on all subsequent bands. The breakpoints are returned as a 2-D matrix of 4 rows and as many columns as images. The first two rows are the original X and Y values. The third row contains the Y values fitted to the estimated segments, and the 4th row contains a 1 if the corresponding point was used as a segment vertex or 0 if not. Any additional fitted bands are appended as rows in the output. Breakpoint fitting assumes that increasing values represent disturbance and decreasing values represent recovery.

See: Kennedy, R.E., Yang, Z. and Cohen, W.B., 2010. Detecting trends in forest disturbance and recovery using yearly Landsat time series: 1. LandTrendr - Temporal segmentation algorithms. Remote Sensing of Environment, 114(12), pp.2897-2910.

Usage	Returns
`ee.Algorithms.TemporalSegmentation.LandTrendr(timeSeries, maxSegments, spikeThreshold, vertexCountOvershoot, preventOneYearRecovery, recoveryThreshold, pvalThreshold, bestModelProportion, minObservationsNeeded)`	Image

Argument	Type	Details
`timeSeries`	ImageCollection	Yearly time-series from which to extract breakpoints. The first band is usedto find breakpoints, and all subsequent bands are fitted using those breakpoints.
`maxSegments`	Integer	Maximum number of segments to be fitted on the time series.
`spikeThreshold`	*Float, default: 0.9*	*Threshold for dampening the spikes (1.0 means no dampening).*
`vertexCountOvershoot`	*Integer, default: 3*	*The initial model can overshoot the maxSegments + 1 vertices by this amount. Later, it will be pruned down to maxSegments + 1.*
`preventOneYearRecovery`	*Boolean, default: false*	*Prevent segments that represent one year recoveries.*
`recoveryThreshold`	*Float, default: 0.25*	*If a segment has a recovery rate faster than 1/recoveryThreshold (in years), then the segment is disallowed.*
`pvalThreshold`	*Float, default: 0.1*	*If the p-value of the fitted model exceeds this threshold, then the current model is discarded and another one is fitted using the Levenberg-Marquardt optimizer.*
`bestModelProportion`	*Float, default: 0.75*	*Allows models with more vertices to be chosen if their p-value is no more than (2 - bestModelProportion) times the p-value of the best model.*
`minObservationsNeeded`	*Integer, default: 6*	*Min observations needed to perform output fitting.*

ee.Algorithms.TemporalSegmentation.LandTrendrFit

Interpolates a time series using a set of LandTrendr breakpoint years. For each input band in the timeSeries, outputs a new 1D array-valued band containing the input values interpolated between the breakpoint times identified by the vertices image. See the LandTrendr Algorithm for more details.

Usage	Returns
`ee.Algorithms.TemporalSegmentation.LandTrendrFit(timeSeries, vertices, spikeThreshold, minObservationsNeeded)`	Image

Argument	Type	Details
`timeSeries`	ImageCollection	Time series to interpolate.
`vertices`	Image	Vertices image. A 1D array of LandTrendr breakpoint years.

Argument	Type	Details
`spikeThreshold`	*Float, default: 0.9*	*Threshold for dampening input spikes (1.0 means no dampening).*
`minObservationsNeeded`	*Integer, default: 6*	*Min observations needed.*

ee.Algorithms.TemporalSegmentation.StructuralChangeBreakpoints

Runs breakpoint detection, similar to R's strucchange::breakpoints function.

Each pixel is fit by a piecewise linear/harmonic model, of the form

Y = A + B * t + C * cos(2 * pi * season(t)) + D * sin(2 * pi * season(t)) + E * cos(4 * pi * season(t)) + F * sin(4 * pi * season(t)) + ...

In this equation, 't' is the start time of the image in the format specified by 'dateFormat', and 'season(t)' is the fractional year of that start time (see the description of dateFormat for details). The maximum order of the harmonic terms is determined by 'seasonalModelOrder'.

The result is an image containing two bands, plus two bands per band in the input:

`tStart`, `tEnd`: each of these holds a 1D array, with one entry per segment in the piecewise linear fit; each entry contains the start time of the first or last images in that segment. By default the values here are in fractional years, for easy use with the coefficients.

`coefs_BANDNAME`: there will be one such output band per input band. Each of these holds a 2D array, with one row per segment. The values in that row are the coefficients of the linear fit for that segment - that is, the values of A, B, C, ... for that segment. As described above, the values here are affected by 'dateFormat'

`.rmse_BANDNAME`: there will be one such output band per input band. This holds a 1D array, with one entry per segment. The value for each segment is the RMSE for the linear fit residuals for that segment.

Usage	Returns
`ee.Algorithms.TemporalSegmentation.StructuralChangeBreakpoints(collection, `*`breakpointBand`*`, `*`seasonalModelOrder`*`, `*`minSpacing`*`, `*`maxBreaks`*`, `*`dateFormat`*`)`	Image

Argument	Type	Details
`collection`	ImageCollection	Collection of images on which to detect breakpoints.
`breakpointBand`	*String, default: null*	*The name of the band to use for breakpoint detection. Optional only if the images have only a single band.*
`seasonalModelOrder`	*Integer, default: 3*	*The order of the harmonic seasonal model.*

Argument	Type	Details
`minSpacing`	*Float, default: 0.15*	*The minimum spacing between breakpoints. If this is between 0 and 1 (exclusive), it will be interpreted as a fraction of the number of images in the collection. Otherwise, it will be interpreted as a number of samples.*
`maxBreaks`	*Integer, default: 0*	*The maximum number of breakpoints.*
`dateFormat`	*Integer, default: 1*	*The time representation to use in the results: 1 = fractional years, 2 = unix time in milliseconds. This affects the values in the tStart and tEnd bands and the 't' values used in the harmonic model. The fractional years used here and in that model are defined as the fractional number of 365.25-day years since 1 Jan 1970.*

ee.Algorithms.TemporalSegmentation.VCT

Vegetation Change Tracker, an automated approach for reconstructing recent forest disturbance history using dense Landsat time series stacks.

The output is a 2D array per pixel containing 6 rows x N years. The output rows contain: input years, VCT landcover mask, magnitude in term of the UD composite, magnitude of distubance in B4, magnitude of distubance in NDVI, magnitude of distubance in dNBR.

See: Huang, C., Goward, S.N., Masek, J.G., Thomas, N., Zhu, Z. and Vogelmann, J.E., 2010. An automated approach for reconstructing recent forest disturbance history using dense Landsat time series stacks. Remote Sensing of Environment, 114(1), pp.183-198.

Usage	Returns
`ee.Algorithms.TemporalSegmentation.VCT(timeSeries, landCover, maxUd, minNdvi, forThrMax, nYears)`	Image

Argument	Type	Details
`timeSeries`	ImageCollection	Collection from which to extract VCT disturbances, containing the bands: B3, B4, B5, B7, thermal, NDVI, DNBR and COMP. This collection is expected to contain 1 image for each year, sorted by time.
`landCover`	ImageCollection	Collection from which to extract VCT masks. This collection is expected to contain 1 image for each image in the timeSeries, sorted by time.
`maxUd`	*Float, default: 4*	*Maximum Z-score composite value for detecting forest.*
`minNdvi`	*Float, default: 0.45*	*Minimum NDVI value for forest.*
`forThrMax`	*Float, default: 3*	*Maximum threshold for forest.*

Argument	Type	Details
nYears	*Integer, default: 30*	*Maximum number of years.*

ee.Algorithms.TemporalSegmentation.Verdet

Vegetation Regeneration and Disturbance Estimates through Time, forest change detection algorithm. This algorithm generates a yearly clear-sky composite from satellite imagery, calculates a spectral vegetation index for each pixel in that composite, spatially segments the vegetation index image into patches, temporally divides the time series into differently sloped segments, and then labels those segments as disturbed, stable, or regenerating. Segmentation at both the spatial and temporal steps are performed using total variation regularization.

The output consists of a 1D array per pixel containing the slope of fitted trend lines. Negative values indicate disturbance and positive values regeneration.

See: Hughes, M.J., Kaylor, S.D. and Hayes, D.J., 2017. Patch-based forest change detection from Landsat time series. Forests, 8(5), p.166.

Usage	Returns
`ee.Algorithms.TemporalSegmentation.Verdet(timeSeries, tolerance, alpha, nRuns)`	Image

Argument	Type	Details
timeSeries	ImageCollection	Collection from which to extract VeRDET scores. This collection is expected to contain 1 image for each year, sorted temporally.
tolerance	*Float, default: 0.0001*	*Convergence tolerance.*
alpha	*Float, default: 0.03333333333333333*	*Regularization parameter for segmentation.*
nRuns	*Integer, default: 100*	*Maximum number of runs for convergence.*

ee.Algorithms.Terrain

Calculates slope, aspect, and a simple hillshade from a terrain DEM.

Expects an image containing either a single band of elevation, measured in meters, or if there's more than one band, one named 'elevation'. Adds output bands named 'slope' and 'aspect' measured in degrees plus an unsigned byte output band named 'hillshade' for visualization. All other bands and metadata are copied from

the input image. The local gradient is computed using the 4-connected neighbors of each pixel, so missing values will occur around the edges of an image.

Usage	Returns
`ee.Algorithms.Terrain(input)`	Image

Argument	Type	Details
`input`	Image	An elevation image, in meters.

ee.Array

Returns an array with the given coordinates.

Usage	Returns
`ee.Array(values, pixelType)`	Array

Argument	Type	Details
`values`	Object	An existing array to cast, or a number/list of numbers/nested list of numbers of any depth to create an array from. For nested lists, all inner arrays at the same depth must have the same length, and numbers may only be present at the deepest level.
`pixelType`	*PixelType, default: null*	*The type of each number in the values argument. If the pixel type is not provided, it will be inferred from the numbers in 'values'. If there aren't any numbers in 'values', this type must be provided.*

ee.Array.abs

On an element-wise basis, computes the absolute value of the input.

Usage	Returns
`Array.abs()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.accum

Accumulates elements of an array along the given axis, by setting each element of the result to the reduction of elements along that axis up to and including the current position. May be used to make a cumulative sum, a monotonically increasing sequence, etc.

Usage	Returns
`Array.accum(axis, reducer)`	Array

Argument	Type	Details
this: `array`	Array	Array to accumulate.
`axis`	Integer	Axis along which to perform the accumulation.
`reducer`	*Reducer, default: null*	*Reducer to accumulate values. Default is SUM, to produce the cumulative sum of each vector along the given axis.*

ee.Array.acos

On an element-wise basis, computes the arc cosine in radians of the input.

Usage	Returns
`Array.acos()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.add

On an element-wise basis, adds the first value to the second.

Usage	Returns
`Array.add(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.and

On an element-wise basis, returns 1 if and only if both values are non-zero.

Usage	Returns
`Array.and(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.argmax

Returns the position, as a list of indices in each array axis, of the maximum value in an array, or null if the array is empty. If there are multiple occurrences of the maximum, returns the position of the first.

Usage	Returns
`Array.argmax()`	List

Argument	Type	Details
this: `array`	Array	

ee.Array.asin

On an element-wise basis, computes the arc sine in radians of the input.

Usage	Returns
`Array.asin()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.atan

On an element-wise basis, computes the arc tangent in radians of the input.

Usage	Returns
`Array.atan()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.atan2

On an element-wise basis, calculates the angle formed by the 2D vector [x, y].

Usage	Returns
`Array.atan2(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.bitCount

On an element-wise basis, calculates the number of one-bits in the 64-bit two's complement binary representation of the input.

Usage	Returns
`Array.bitCount()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.bitsToArray

Convert the bits of an integer to an Array. The array has as many elements as the position of the highest set bit, or a single 0 for a value of 0.

Usage	Returns
`ee.Array.bitsToArray(input)`	Array

Argument	Type	Details
`input`	Number	

ee.Array.bitwiseAnd

On an element-wise basis, calculates the bitwise AND of the input values.

Usage	Returns
`Array.bitwiseAnd(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.bitwiseNot

On an element-wise basis, calculates the bitwise NOT of the input, in the smallest signed integer type that can hold the input.

Usage	Returns
`Array.bitwiseNot()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.bitwiseOr

On an element-wise basis, calculates the bitwise OR of the input values.

Usage	Returns
`Array.bitwiseOr(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.bitwiseXor

On an element-wise basis, calculates the bitwise XOR of the input values.

Usage	Returns
`Array.bitwiseXor(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.byte

On an element-wise basis, casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Array.byte()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.cat

Concatenates multiple arrays into a single array along the given axis. Each array must have the same dimensionality and the same length on all axes except the concatenation axis.

Usage	Returns
`ee.Array.cat(arrays, axis)`	Array

Argument	Type	Details
`arrays`	List	Arrays to concatenate.
`axis`	*Integer, default: 0*	*Axis to concatenate along.*

ee.Array.cbrt

On an element-wise basis, computes the cubic root of the input.

Usage	Returns
`Array.cbrt()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.ceil

On an element-wise basis, computes the smallest integer greater than or equal to the input.

Usage	Returns
`Array.ceil()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.cos

On an element-wise basis, computes the cosine of the input in radians.

Usage	Returns
`Array.cos()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.cosh

On an element-wise basis, computes the hyperbolic cosine of the input.

Usage	Returns
`Array.cosh()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.cut

Cut an array along one or more axes.

Usage	Returns
`Array.cut(position)`	Array

Argument	Type	Details
this: `array`	Array	The array to cut.
`position`	List	Cut an array along one or more axes. The positions args specifies either a single value for each axis of the array, or -1, indicating the whole axis. The output will be an array that has the same dimensions as the input, with a length of 1 on each axis that was not -1 in the positions array.

ee.Array.digamma

On an element-wise basis, computes the digamma function of the input.

Usage	Returns
`Array.digamma()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.divide

On an element-wise basis, divides the first value by the second, returning 0 for division by 0.

Usage	Returns
`Array.divide(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.dotProduct

Compute the dot product between two 1-D arrays.

Usage	Returns
`Array.dotProduct(array2)`	Number

Argument	Type	Details
this: `array1`	Array	The first 1-D array.
`array2`	Array	The second 1-D array.

ee.Array.double

On an element-wise basis, casts the input value to a 64-bit float.

Usage	Returns
`Array.double()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.eigen

Computes the real eigenvectors and eigenvalues of a square 2D array of A rows and A columns. Returns an array with A rows and A+1 columns, where each row contains an eigenvalue in the first column, and the corresponding eigenvector in the remaining A columns. The rows are sorted by eigenvalue, in descending order.

This implementation uses DecompositionFactory.eig() from https://ejml.org.

Usage	Returns
`Array.eigen()`	Array

Argument	Type	Details
this: `input`	Array	A square, 2D array from which to compute the eigenvalue decomposition.

ee.Array.eq

On an element-wise basis, returns 1 if and only if the first value is equal to the second.

Usage	Returns
`Array.eq(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.erf

On an element-wise basis, computes the error function of the input.

Usage	Returns
`Array.erf()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.erfInv

On an element-wise basis, computes the inverse error function of the input.

Usage	Returns
`Array.erfInv()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.erfc

On an element-wise basis, computes the complementary error function of the input.

Usage	Returns
`Array.erfc()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.erfcInv

On an element-wise basis, computes the inverse complementary error function of the input.

Usage	Returns
`Array.erfcInv()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.exp

On an element-wise basis, computes the Euler's number e raised to the power of the input.

Usage	Returns
`Array.exp()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.first

On an element-wise basis, selects the value of the first value.

Usage	Returns
`Array.first(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.firstNonZero

On an element-wise basis, selects the first value if it is non-zero, and the second value otherwise.

Usage	Returns
`Array.firstNonZero(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.float

On an element-wise basis, casts the input value to a 32-bit float.

Usage	Returns
`Array.float()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.floor

On an element-wise basis, computes the largest integer less than or equal to the input.

Usage	Returns
`Array.floor()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.gamma

On an element-wise basis, computes the gamma function of the input.

Usage	Returns
`Array.gamma()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.gammainc

On an element-wise basis, calculates the regularized lower incomplete Gamma function γ(x,a).

Usage	Returns
`Array.gammainc(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.get

Extracts the value at the given position from the input array.

Usage	Returns
`Array.get(position)`	Number

Argument	Type	Details
this: `array`	Array	The array to extract from.
`position`	List	The coordinates of the element to get.

ee.Array.gt

On an element-wise basis, returns 1 if and only if the first value is greater than the second.

Usage	Returns
`Array.gt(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.gte

On an element-wise basis, returns 1 if and only if the first value is greater than or equal to the second.

Usage	Returns
`Array.gte(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.hypot

On an element-wise basis, calculates the magnitude of the 2D vector [x, y].

Usage	Returns
`Array.hypot(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.identity

Creates a 2D identity matrix of the given size.

Usage	Returns
`ee.Array.identity(size)`	Array

Argument	Type	Details
`size`	Integer	The length of each axis.

ee.Array.int

On an element-wise basis, casts the input value to a signed 32-bit integer.

Usage	Returns
`Array.int()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.int16

On an element-wise basis, casts the input value to a signed 16-bit integer.

Usage	Returns
`Array.int16()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.int32

On an element-wise basis, casts the input value to a signed 32-bit integer.

Usage	Returns
`Array.int32()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.int64

On an element-wise basis, casts the input value to a signed 64-bit integer.

Usage	Returns
`Array.int64()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.int8

On an element-wise basis, casts the input value to a signed 8-bit integer.

Usage	Returns
`Array.int8()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.lanczos

On an element-wise basis, computes the Lanczos approximation of the input.

Usage	Returns
`Array.lanczos()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.leftShift

On an element-wise basis, calculates the left shift of v1 by v2 bits.

Usage	Returns
`Array.leftShift(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.length

Returns a 1-D ee.Array containing the length of each dimension of the given ee.Array.

Usage	Returns
`Array.length()`	Array

Argument	Type	Details
this: `array`	Array	The array from which to extract the axis lengths.

ee.Array.log

On an element-wise basis, computes the natural logarithm of the input.

Usage	Returns
`Array.log()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.log10

On an element-wise basis, computes the base-10 logarithm of the input.

Usage	Returns
`Array.log10()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.long

On an element-wise basis, casts the input value to a signed 64-bit integer.

Usage	Returns
`Array.long()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.lt

On an element-wise basis, returns 1 if and only if the first value is less than the second.

Usage	Returns
`Array.lt(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.lte

On an element-wise basis, returns 1 if and only if the first value is less than or equal to the second.

Usage	Returns
`Array.lte(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.mask

Creates a subarray by slicing out each position in an input array that is parallel to a non-zero element of the given mask array.

Usage	Returns
`Array.mask(mask)`	Array

Argument	Type	Details
this: `input`	Array	Array to mask.
`mask`	Array	Mask array.

ee.Array.matrixCholeskyDecomposition

Calculates the Cholesky decomposition of a matrix. The Cholesky decomposition is a decomposition into the form L * L' where L is a lower triangular matrix. The input must be a symmetric positive-definite matrix. Returns a dictionary with 1 entry named 'L'.

Usage	Returns
`Array.matrixCholeskyDecomposition()`	Dictionary

Argument	Type	Details
this: `array`	Array	The array to decompose.

ee.Array.matrixDeterminant

Computes the determinant of the matrix.

Usage	Returns
`Array.matrixDeterminant()`	Number

Argument	Type	Details
this: `input`	Array	The array to compute on.

ee.Array.matrixDiagonal

Computes the diagonal of the matrix in a single column.

Usage	Returns
`Array.matrixDiagonal()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.matrixFnorm

Computes the Frobenius norm of the matrix.

Usage	Returns
`Array.matrixFnorm()`	Number

Argument	Type	Details
this: `input`	Array	The array to compute on.

ee.Array.matrixInverse

Computes the inverse of the matrix.

Usage	Returns
`Array.matrixInverse()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.matrixLUDecomposition

Calculates the LU matrix decomposition such that P×input=L×U, where L is lower triangular (with unit diagonal terms), U is upper triangular and P is a partial pivot permutation matrix. The input matrix must be square. Returns a dictionary with entries named 'L', 'U' and 'P'.

Usage	Returns
`Array.matrixLUDecomposition()`	Dictionary

Argument	Type	Details
this: `array`	Array	The array to decompose.

ee.Array.matrixMultiply

Returns the matrix multiplication A * B.

Usage	Returns
`Array.matrixMultiply(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.matrixPseudoInverse

Computes the Moore-Penrose pseudoinverse of the matrix.

Usage	Returns
`Array.matrixPseudoInverse()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.matrixQRDecomposition

Calculates the QR-decomposition of a matrix into two matrices Q and R such that input = QR, where Q is orthogonal, and R is upper triangular. Returns a dictionary with entries named 'Q' and 'R'.

Usage	Returns
`Array.matrixQRDecomposition()`	Dictionary

Argument	Type	Details
this: `array`	Array	The array to decompose.

ee.Array.matrixSingularValueDecomposition

Calculates the Singular Value Decomposition of the input matrix into U×S×V', such that U and V are orthogonal and S is diagonal. Returns a dictionary with entries named 'U', 'S' and 'V'.

Usage	Returns
`Array.matrixSingularValueDecomposition()`	Dictionary

Argument	Type	Details
this: `array`	Array	The array to decompose.

ee.Array.matrixSolve

Solves for x in the matrix equation A * x = B, finding a least-squares solution if A is overdetermined.

Usage	Returns
`Array.matrixSolve(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.matrixToDiag

Computes a square diagonal matrix from a single column matrix.

Usage	Returns
`Array.matrixToDiag()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.matrixTrace

Computes the trace of the matrix.

Usage	Returns
`Array.matrixTrace()`	Number

Argument	Type	Details
this: `input`	Array	The array to compute on.

ee.Array.matrixTranspose

Transposes two dimensions of an array.

Usage	Returns
`Array.matrixTranspose(`*`axis1`*`, `*`axis2`*`)`	Array

Argument	Type	Details
this: `array`	Array	Array to transpose.
`axis1`	*Integer, default: 0*	*First axis to swap.*
`axis2`	*Integer, default: 1*	*Second axis to swap.*

ee.Array.max

On an element-wise basis, selects the maximum of the first and second values.

Usage	Returns
`Array.max(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.min

On an element-wise basis, selects the minimum of the first and second values.

Usage	Returns
`Array.min(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.mod

On an element-wise basis, calculates the remainder of the first value divided by the second.

Usage	Returns
`Array.mod(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.multiply

On an element-wise basis, multiplies the first value by the second.

Usage	Returns
`Array.multiply(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.neq

On an element-wise basis, returns 1 if and only if the first value is not equal to the second.

Usage	Returns
`Array.neq(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.not

On an element-wise basis, returns 0 if the input is non-zero, and 1 otherwise.

Usage	Returns
`Array.not()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.or

On an element-wise basis, returns 1 if and only if either input value is non-zero.

Usage	Returns
`Array.or(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.pad

Pad an array to a given length. The pad value will be repeatedly appended to the array to extend it to given length along each axis. If the array is already as large or larger than a given length, it will remain unchanged along that axis.

Usage	Returns
`Array.pad(lengths, pad)`	Array

Argument	Type	Details
this: `array`	Array	Array to pad.
`lengths`	List	A list of new lengths for each axis.
`pad`	*Number, default: 0*	*The value with which to pad the array.*

ee.Array.pow

On an element-wise basis, raises the first value to the power of the second.

Usage	Returns
`Array.pow(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.

Argument	Type	Details
`right`	Array	The right-hand value.

ee.Array.project

Projects an array to a lower dimensional space by specifying the axes to retain. Dropped axes must be at most length 1.

Usage	Returns
`Array.project(axes)`	Array

Argument	Type	Details
this: `array`	Array	Array to project.
`axes`	List	The axes to project onto. Other axes will be discarded, and must be at most length 1.

ee.Array.reduce

Apply a reducer to an array by collapsing all the input values along each specified axis into a single output value computed by the reducer.

The output always has the same dimensionality as the input, and the individual axes are affected as follows:

- The axes specified in the 'axes' parameter have their length reduced to 1 (by applying the reducer).
- If the reducer has multiple inputs or multiple outputs, the axis specified in 'fieldAxis' will be used to provide the reducer's inputs and store the reducer's outputs.
- All other axes are unaffected (independent reductions are performed).

Usage	Returns
`Array.reduce(reducer, axes, `*`fieldAxis`*`)`	Array

Argument	Type	Details
this: `array`	Array	The array.
`reducer`	Reducer	The reducer to apply. Each of its outputs must be a number, not an array or other type.

Argument	Type	Details
`axes`	List	The list of axes over which to reduce. The output will have a length of 1 in all these axes.
`fieldAxis`	*Integer, default: null*	*The axis to use as the reducer's input and output fields. Only required if the reducer has multiple inputs or multiple outputs, in which case the axis must have length equal to the number of reducer inputs, and in the result it will have length equal to the number of reducer outputs.*

ee.Array.repeat

Repeats the array along the given axis. The result will have the shape of the input, except length along the repeated axis will be multiplied by the given number of copies.

Usage	Returns
`Array.repeat(`*`axis`*`, `*`copies`*`)`	Array

Argument	Type	Details
this: `array`	Array	Array to repeat.
`axis`	*Integer, default: 0*	*The axis along which to repeat the array.*
`copies`	*Integer, default: 2*	*The number of copies of this array to concatenate along the given axis.*

ee.Array.reshape

Reshapes an array to a new list of dimension lengths.

Usage	Returns
`Array.reshape(shape)`	Array

Argument	Type	Details
this: `array`	Array	Array to reshape.
`shape`	Array	New shape to which arrays are converted. If one component of the shape is the special value -1, the size of that dimension is computed so that the total size remains constant. In particular, a shape of [-1] flattens into 1-D. At most one component of shape can be -1.

ee.Array.rightShift

On an element-wise basis, calculates the signed right shift of v1 by v2 bits.

Usage	Returns
`Array.rightShift(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.round

On an element-wise basis, computes the integer nearest to the input.

Usage	Returns
`Array.round()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.short

On an element-wise basis, casts the input value to a signed 16-bit integer.

Usage	Returns
`Array.short()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.signum

On an element-wise basis, computes the signum function (sign) of the input; zero if the input is zero, 1 if the input is greater than zero, -1 if the input is less than zero.

Usage	Returns
`Array.signum()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.sin

On an element-wise basis, computes the sine of the input in radians.

Usage	Returns
`Array.sin()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.sinh

On an element-wise basis, computes the hyperbolic sine of the input.

Usage	Returns
`Array.sinh()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.slice

Creates a subarray by slicing out each position along the given axis from the 'start' (inclusive) to 'end' (exclusive) by increments of 'step'. The result will have as many dimensions as the input, and the same length in

all directions except the slicing axis, where the length will be the number of positions from 'start' to 'end' by 'step' that are in range of the input array's length along 'axis'. This means the result can be length 0 along the given axis if start=end, or if the start or end values are entirely out of range.

Usage	Returns
`Array.slice(`*`axis`*`, `*`start`*`, `*`end`*`, `*`step`*`)`	Array

Argument	Type	Details
this: `array`	Array	Array to slice.
`axis`	*Integer, default: 0*	*The axis to slice on.*
`start`	*Integer, default: 0*	*The coordinate of the first slice (inclusive) along 'axis'. Negative numbers are used to position the start of slicing relative to the end of the array, where -1 starts at the last position on the axis, -2 starts at the next to last position, etc.*
`end`	*Integer, default: null*	*The coordinate (exclusive) at which to stop taking slices. By default this will be the length of the given axis. Negative numbers are used to position the end of slicing relative to the end of the array, where -1 will exclude the last position, -2 will exclude the last two positions, etc.*
`step`	*Integer, default: 1*	*The separation between slices along 'axis'; a slice will be taken at each whole multiple of 'step' from 'start' (inclusive) to 'end' (exclusive). Must be positive.*

ee.Array.sort

Sorts elements of the array along one axis.

Usage	Returns
`Array.sort(`*`keys`*`)`	Array

Argument	Type	Details
this: `array`	Array	Array image to sort.
`keys`	*Array, default: null*	*Optional keys to sort by. If not provided, the values are used as the keys. The keys can only have multiple elements along one axis, which determines the direction to sort in.*

ee.Array.sqrt

On an element-wise basis, computes the square root of the input.

Usage	Returns
`Array.sqrt()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.subtract

On an element-wise basis, subtracts the second value from the first.

Usage	Returns
`Array.subtract(right)`	Array

Argument	Type	Details
this: `left`	Array	The left-hand value.
`right`	Array	The right-hand value.

ee.Array.tan

On an element-wise basis, computes the tangent of the input in radians.

Usage	Returns
`Array.tan()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.tanh

On an element-wise basis, computes the hyperbolic tangent of the input.

Usage	Returns
`Array.tanh()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toByte

On an element-wise basis, casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Array.toByte()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toDouble

On an element-wise basis, casts the input value to a 64-bit float.

Usage	Returns
`Array.toDouble()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toFloat

On an element-wise basis, casts the input value to a 32-bit float.

Usage	Returns
`Array.toFloat()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toInt

On an element-wise basis, casts the input value to a signed 32-bit integer.

Usage	Returns
`Array.toInt()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toInt16

On an element-wise basis, casts the input value to a signed 16-bit integer.

Usage	Returns
`Array.toInt16()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toInt32

On an element-wise basis, casts the input value to a signed 32-bit integer.

Usage	Returns
`Array.toInt32()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toInt64

On an element-wise basis, casts the input value to a signed 64-bit integer.

Usage	Returns
`Array.toInt64()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toInt8

On an element-wise basis, casts the input value to a signed 8-bit integer.

Usage	Returns
`Array.toInt8()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toList

Turns an Array into a list of lists of numbers.

Usage	Returns
`Array.toList()`	List

Argument	Type	Details
this: `array`	Array	Array to convert.

ee.Array.toLong

On an element-wise basis, casts the input value to a signed 64-bit integer.

Usage	Returns
`Array.toLong()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toShort

On an element-wise basis, casts the input value to a signed 16-bit integer.

Usage	Returns
`Array.toShort()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toUint16

On an element-wise basis, casts the input value to an unsigned 16-bit integer.

Usage	Returns
`Array.toUint16()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toUint32

On an element-wise basis, casts the input value to an unsigned 32-bit integer.

Usage	Returns
`Array.toUint32()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.toUint8

On an element-wise basis, casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Array.toUint8()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.transpose

Transposes two dimensions of an array.

Usage	Returns
`Array.transpose(`*`axis1`*`, `*`axis2`*`)`	Array

Argument	Type	Details
this: `array`	Array	Array to transpose.
`axis1`	*Integer, default: 0*	*First axis to swap.*
`axis2`	*Integer, default: 1*	*Second axis to swap.*

ee.Array.trigamma

On an element-wise basis, computes the trigamma function of the input.

Usage	Returns
`Array.trigamma()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.uint16

On an element-wise basis, casts the input value to an unsigned 16-bit integer.

Usage	Returns
`Array.uint16()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.uint32

On an element-wise basis, casts the input value to an unsigned 32-bit integer.

Usage	Returns
`Array.uint32()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Array.uint8

On an element-wise basis, casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Array.uint8()`	Array

Argument	Type	Details
this: `input`	Array	The input array.

ee.Blob

Loads a Blob from a Google Cloud Storage URL.

Usage	Returns
`ee.Blob(url)`	Blob

Argument	Type	Details
`url`	String	The Blob's Google Cloud Storage URL.

ee.Blob.string

Returns the contents of the blob as a String.

Usage	Returns
`Blob.string(`*`encoding`*`)`	String

Argument	Type	Details
this: `blob`	Blob	
`encoding`	*String, default: null*	

ee.Blob.url

Returns the Blob's Google Cloud Storage URL.

Usage	Returns
`Blob.url()`	String

Argument	Type	Details
this: `blob`	Blob	

ee.Classifier.amnhMaxent

Creates a Maximum Entropy classifier. Maxent is used to model species distribution probabilities using environmental data for locations of known presence and for a large number of 'background' locations. For more information and to cite, see: https://biodiversityinformatics.amnh.org/open_source/maxent/ and the reference publication: Phillips, et. al., 2004 A maximum entropy approach to species distribution modeling, Proceedings of the Twenty-First International Conference on Machine Learning. The output is a single band named 'probability', containing the modeled probability, and an additional band named 'clamp' when the 'writeClampGrid' argument is true.

Usage	Returns
`ee.Classifier.amnhMaxent(`*`categoricalNames`*`, `*`outputFormat`*`, `*`autoFeature`*`, `*`linear`*`, `*`quadratic`*`, `*`product`*`, `*`threshold`*`, `*`hinge`*`, `*`hingeThreshold`*`, `*`l2lqThreshold`*`, `*`lq2lqptThreshold`*`, `*`addSamplesToBackground`*`, `*`addAllSamplesToBackground`*`, `*`betaMultiplier`*`, `*`betaHinge`*`, `*`betaLqp`*`, `*`betaCategorical`*`, `*`betaThreshold`*`, `*`extrapolate`*`, `*`doClamp`*`, `*`writeClampGrid`*`, `*`randomTestPoints`*`, `*`seed`*`)`	Classifier

Argument	Type	Details
`categoricalNames`	*List, default: null*	*A list of the names of the categorical inputs. Any inputs not listed in this argument are considered to be continuous.*
`outputFormat`	*String, default: "cloglog"*	*Representation of probabilities in output.*
`autoFeature`	*Boolean, default: true*	*Automatically select which feature classes to use, based on number of training samples.*
`linear`	*Boolean, default: true*	*Allow linear features to be used. Ignored when autofeature is true.*
`quadratic`	*Boolean, default: true*	*Allow quadratic features to be used. Ignored when autofeature is true.*
`product`	*Boolean, default: true*	*Allow product features to be used. Ignored when autofeature is true.*
`threshold`	*Boolean, default: false*	*Allow threshold features to be used. Ignored when autofeature is true.*
`hinge`	*Boolean, default: true*	*Allow hinge features to be used. Ignored when autofeature is true.*
`hingeThreshold`	*Integer, default: 15*	*Number of samples at which hinge features start being used. Ignored when autofeature is false.*
`l2lqThreshold`	*Integer, default: 10*	*Number of samples at which quadratic features start being used. Ignored when autofeature is false.*
`lq2lqptThreshold`	*Integer, default: 80*	*Number of samples at which product and threshold features start being used. Ignored when autofeature is false.*
`addSamplesToBackground`	*Boolean, default: true*	*Add to the background any sample for which has a combination of environmental values that isn't already present in the background.*
`addAllSamplesToBackground`	*Boolean, default: false*	*Add all samples to the background, even if they have combinations of environmental values that are already present in the background.*
`betaMultiplier`	*Float, default: 1*	*Regularization multiplier. Multiply all automatic regularization parameters by this number. A higher number gives a more spread-out distribution.*
`betaHinge`	*Float, default: -1*	*Regularization parameter to be applied to all hinge features; negative value enables automatic setting.*
`betaLqp`	*Float, default: -1*	*Regularization parameter to be applied to all linear, quadratic and product features; negative value enables automatic setting.*
`betaCategorical`	*Float, default: -1*	*Regularization parameter to be applied to all categorical features; negative value enables automatic setting.*
`betaThreshold`	*Float, default: -1*	*Regularization parameter to be applied to all threshold features; negative value enables automatic setting.*

Argument	Type	Details
`extrapolate`	*Boolean, default: true*	*Extrapolate. Predict to regions of environmental space outside the limits encountered during training.*
`doClamp`	*Boolean, default: true*	*Apply clamping to output.*
`writeClampGrid`	*Boolean, default: true*	*Adds a band to the output ('clamp') showing the spatial distribution of clamping. At each point, the value is the absolute difference between prediction values with and without clamping.*
`randomTestPoints`	*Integer, default: 0*	*Random test percentage. The percentage of training points to hold aside as test points, used to compute AUX, omission, etc.*
`seed`	*Long, default: 0*	*A seed used when generating random numbers.*

ee.Classifier.confusionMatrix

Computes a 2D confusion matrix for a classifier based on its training data (ie: resubstitution error). Axis 0 of the matrix corresponds to the input classes, and axis 1 corresponds to the output classes. The rows and columns start at class 0 and increase sequentially up to the maximum class value, so some rows or columns might be empty if the input classes aren't 0-based or sequential.

Usage	Returns
`Classifier.confusionMatrix()`	ConfusionMatrix

Argument	Type	Details
this: `classifier`	Classifier	The classifier to use.

ee.Classifier.decisionTree

Creates a classifier that applies the given decision tree.

Usage	Returns
`ee.Classifier.decisionTree(treeString)`	Classifier

Argument	Type	Details
`treeString`	String	The decision tree, specified in the text format generated by R and other similar tools.

ee.Classifier.decisionTreeEnsemble

Creates a classifier that applies the given decision trees.

Usage	Returns
`ee.Classifier.decisionTreeEnsemble(treeStrings)`	Classifier

Argument	Type	Details
`treeStrings`	List	The decision trees, specified in the text format generated by R and other similar tools. Each item in the list should contain one or more trees in text format.

ee.Classifier.explain

Describe the results of a trained classifier.

Usage	Returns
`Classifier.explain()`	Dictionary

Argument	Type	Details
this: `classifier`	Classifier	The classifier to describe.

ee.Classifier.libsvm

Creates an empty Support Vector Machine classifier.

Usage	Returns
`ee.Classifier.libsvm(`*`decisionProcedure`*`, `*`svmType`*`, `*`kernelType`*`, `*`shrinking`*`, `*`degree`*`, `*`gamma`*`, `*`coef0`*`, `*`cost`*`, `*`nu`*`, `*`terminationEpsilon`*`, `*`lossEpsilon`*`, `*`oneClass`*`)`	Classifier

Argument	Type	Details
`decisionProcedure`	*String, default: "Voting"*	*The decision procedure to use for classification. Either 'Voting' or 'Margin'. Not used for regression.*
`svmType`	*String, default: "C_SVC"*	*The SVM type. One of `C_SVC`, `NU_SVC`, `ONE_CLASS`, `EPSILON_SVR` or `NU_SVR`.*
`kernelType`	*String, default: "LINEAR"*	*The kernel type. One of LINEAR (u'×v), POLY ((γ×u'×v + $coef_0$)^{degree}), RBF (exp(-γ×|u-v|²)) or SIGMOID (tanh(γ×u'×v + $coef_0$)).*
`shrinking`	*Boolean, default: true*	*Whether to use shrinking heuristics.*
`degree`	*Integer, default: null*	*The degree of polynomial. Valid for POLY kernels.*
`gamma`	*Float, default: null*	*The gamma value in the kernel function. Defaults to the reciprocal of the number of features. Valid for POLY, RBF and SIGMOID kernels.*
`coef0`	*Float, default: null*	*The $coef_0$ value in the kernel function. Defaults to 0. Valid for POLY and SIGMOID kernels.*
`cost`	*Float, default: null*	*The cost (C) parameter. Defaults to 1. Only valid for C-SVC, epsilon-SVR, and nu-SVR.*
`nu`	*Float, default: null*	*The nu parameter. Defaults to 0.5. Only valid for nu-SVC, one-class SVM, and nu-SVR.*
`terminationEpsilon`	*Float, default: null*	*The termination criterion tolerance (e). Defaults to 0.001. Only valid for epsilon-SVR.*
`lossEpsilon`	*Float, default: null*	*The epsilon in the loss function (p). Defaults to 0.1. Only valid for epsilon-SVR.*
`oneClass`	*Integer, default: null*	*The class of the training data on which to train in a one-class SVM. Defaults to 0. Only valid for one-class SVM. Possible values are 0 and 1. The classifier output is binary (0/1) and will match this class value for the data determined to be in the class.*

ee.Classifier.load

Creates a Classifier.

Usage	Returns
`ee.Classifier.load(id)`	Classifier

Argument	Type	Details
`id`	String	The Classifier's Asset ID.

ee.Classifier.minimumDistance

Creates a minimum distance classifier for the given distance metric. In CLASSIFICATION mode, the nearest class is returned. In REGRESSION mode, the distance to the nearest class center is returned. In RAW mode, the distance to every class center is returned.

Usage	Returns
`ee.Classifier.minimumDistance(`*`metric`*`, `*`kNearest`*`)`	Classifier

Argument	Type	Details
`metric`	*String, default: "euclidean"*	*The distance metric to use. Options are:* • *'euclidean' - euclidean distance from the unnormalized class mean.* • *'cosine' - spectral angle from the unnormalized class mean.* • *'mahalanobis' - Mahalanobis distance from the class mean.* • *'manhattan' - Manhattan distance from the unnormalized class mean.*
`kNearest`	*Integer, default: 1*	*If greater than 1, the result will contain an array of the k nearest neighbors or distances, based on the output mode setting. if kNearest is greater than the total number of classes, it will be set equal to the number of classes.*

ee.Classifier.mode

Returns the classifier mode: `CLASSIFICATION`, `REGRESSION`, `PROBABILITY`, `MULTIPROBABILITY`, `RAW` or `RAW_REGRESSION`.

Usage	Returns
`Classifier.mode()`	String

Argument	Type	Details
this: `classifier`	Classifier	

ee.Classifier.schema

Returns the names of the inputs used by this classifier, or null if this classifier has not had any training data added yet.

Usage	Returns
`Classifier.schema()`	List

Argument	Type	Details
this: `classifier`	Classifier	

ee.Classifier.setOutputMode

Sets the output mode.

Usage	Returns
`Classifier.setOutputMode(mode)`	Classifier

Argument	Type	Details
this: `classifier`	Classifier	An input classifier.
`mode`	String	The output mode. One of: • CLASSIFICATION (default): The output is the class number. • REGRESSION: The output is the result of standard regression. • PROBABILITY: The output is the probability that the classification is correct. • MULTIPROBABILITY: The output is an array of probabilities that each class is correct ordered by classes seen. • RAW: The output is an array of the internal representation of the classification process. For example, the raw votes in multi-decision tree models. • RAW_REGRESSION: The output is an array of the internal representation of the regression process. For example, the raw predictions of multiple regression trees. Not all classifiers support modes other than CLASSIFICATION.

ee.Classifier.smileCart

Creates an empty CART classifier. See:

"Classification and Regression Trees,"

L. Breiman, J. Friedman, R. Olshen, C. Stone

Chapman and Hall, 1984.

Usage	Returns
`ee.Classifier.smileCart(`*`maxNodes`*`, `*`minLeafPopulation`*`)`	Classifier

Argument	Type	Details
`maxNodes`	*Integer, default: null*	*The maximum number of leaf nodes in each tree. If unspecified, defaults to no limit.*
`minLeafPopulation`	*Integer, default: 1*	*Only create nodes whose training set contains at least this many points.*

ee.Classifier.smileGradientTreeBoost

Creates an empty Gradient Tree Boost classifier.

Usage	Returns
`ee.Classifier.smileGradientTreeBoost(numberOfTrees, `*`shrinkage`*`, `*`samplingRate`*`, `*`maxNodes`*`, `*`loss`*`, `*`seed`*`)`	Classifier

Argument	Type	Details
`numberOfTrees`	Integer	The number of decision trees to create.
`shrinkage`	*Float, default: 0.005*	*The shrinkage parameter in (0, 1] controls the learning rate of procedure.*
`samplingRate`	*Float, default: 0.7*	*The sampling rate for stochastic tree boosting.*
`maxNodes`	*Integer, default: null*	*The maximum number of leaf nodes in each tree. If unspecified, defaults to no limit.*
`loss`	*String, default: "LeastAbsoluteDeviation"*	*Loss function for regression. One of: LeastSquares, LeastAbsoluteDeviation, Huber.*
`seed`	*Integer, default: 0*	*The randomization seed.*

ee.Classifier.smileKNN

Creates an empty kNN classifier.

The k-nearest neighbor algorithm (k-NN) is a method for classifying objects by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small, typically odd).

Usage	Returns
`ee.Classifier.smileKNN(k, searchMethod, metric)`	Classifier

Argument	Type	Details
k	*Integer, default: 1*	*The number of neighbors for classification.*
`searchMethod`	*String, default: "AUTO"*	*Search method. The following are valid [AUTO, LINEAR_SEARCH, KD_TREE, COVER_TREE]. AUTO Will choose between KD_TREE and COVER_TREE depending on the dimension count. Results may vary between the different search methods for distance ties and probability values. Since performance and results may vary consult with SMILE's documentation and other literature.*
`metric`	*String, default: "EUCLIDEAN"*	*The distance metric to use. NOTE: KD_TREE (and AUTO for low dimensions) will not use the metric selected. Options are: 'EUCLIDEAN' - euclidean distance. 'MAHALANOBIS' - Mahalanobis distance. 'MANHATTAN' - Manhattan distance.*

ee.Classifier.smileNaiveBayes

Creates an empty Naive Bayes classifier. This classifier assumes that the feature vector consists of positive integers, and negative inputs are discarded.

Usage	Returns
`ee.Classifier.smileNaiveBayes(lambda)`	Classifier

Argument	Type	Details
`lambda`	*Float, default: 0.000001*	*A smoothing lambda. Used to avoid assigning zero probability to classes not seen during training, instead using lambda / (lambda * nFeatures).*

ee.Classifier.smileRandomForest

Creates an empty Random Forest classifier.

Usage	Returns
`ee.Classifier.smileRandomForest(numberOfTrees, variablesPerSplit, minLeafPopulation, bagFraction, maxNodes, seed)`	Classifier

Argument	Type	Details
`numberOfTrees`	Integer	The number of decision trees to create.
`variablesPerSplit`	*Integer, default: null*	*The number of variables per split. If unspecified, uses the square root of the number of variables.*
`minLeafPopulation`	*Integer, default: 1*	*Only create nodes whose training set contains at least this many points.*
`bagFraction`	*Float, default: 0.5*	*The fraction of input to bag per tree.*
`maxNodes`	*Integer, default: null*	*The maximum number of leaf nodes in each tree. If unspecified, defaults to no limit.*
`seed`	*Integer, default: 0*	*The randomization seed.*

ee.Classifier.spectralRegion

Creates a classifier that tests if its inputs lie within a polygon defined by a set of coordinates in an arbitrary 2D coordinate system. Each input to be classified must have 2 values (e.g.: images must have 2 bands). The result will be 1 wherever the input values are contained within the given polygon and 0 otherwise.

Usage	Returns
`ee.Classifier.spectralRegion(coordinates, schema)`	Classifier

Argument	Type	Details
`coordinates`	List	The coordinates of the polygon, as a list of rings. Each ring is a list of coordinate pairs (e.g.: [u1, v1, u2, v2, ..., uN, vN]). No edge may intersect any other edge. The resulting classification will be a 1 wherever the input values are within the interior of the given polygon, that is, an odd number of polygon edges must be crossed to get outside the polygon and 0 otherwise.
`schema`	*List, default: null*	*The classifier's schema. A list of band or property names that the classifier will operate on. Since this classifier doesn't undergo a training step, these have to be specified manually. Defaults to ['u', 'v'].*

ee.Classifier.train

Trains the classifier on a collection of features, using the specified numeric properties of each feature as training data. The geometry of the features is ignored.

Usage	Returns
`Classifier.train(features, classProperty, inputProperties, subsampling, subsamplingSeed)`	Classifier

Argument	Type	Details
this: `classifier`	Classifier	An input classifier.
`features`	FeatureCollection	The collection to train on.
`classProperty`	String	The name of the property containing the class value. Each feature must have this property, and its value must be numeric.
`inputProperties`	*List, default: null*	*The list of property names to include as training data. Each feature must have all these properties, and their values must be numeric. This argument is optional if the input collection contains a 'band_order' property, (as produced by Image.sample).*
`subsampling`	*Float, default: 1*	*An optional subsampling factor, within (0, 1].*
`subsamplingSeed`	*Integer, default: 0*	*A randomization seed to use for subsampling.*

ee.Clusterer.schema

Returns the names of the inputs used by this Clusterer, or null if this Clusterer has not had any training data added yet.

Usage	Returns
`Clusterer.schema()`	List

Argument	Type	Details
this: `clusterer`	Clusterer	

ee.Clusterer.train

Trains the Clusterer on a collection of features, using the specified numeric properties of each feature as training data. The geometry of the features is ignored.

Usage	Returns
`Clusterer.train(features, inputProperties, subsampling, subsamplingSeed)`	Clusterer

Argument	Type	Details
this: `clusterer`	Clusterer	An input Clusterer.
`features`	FeatureCollection	The collection to train on.
`inputProperties`	*List, default: null*	*The list of property names to include as training data. Each feature must have all these properties, and their values must be numeric. This argument is optional if the input collection contains a 'band_order' property, (as produced by Image.sample).*
`subsampling`	*Float, default: 1*	*An optional subsampling factor, within (0, 1].*
`subsamplingSeed`	*Integer, default: 0*	*A randomization seed to use for subsampling.*

ee.Clusterer.wekaCascadeKMeans

Cascade simple k-means, selects the best k according to the Calinski-Harabasz criterion. For more information see:

Calinski, T. and J. Harabasz. 1974. A dendrite method for cluster analysis. Commun. Stat. 3: 1-27.

Usage	Returns
`ee.Clusterer.wekaCascadeKMeans(minClusters, maxClusters, restarts, manual, init, distanceFunction, maxIterations)`	Clusterer

Argument	Type	Details
`minClusters`	*Integer, default: 2*	*Min number of clusters.*
`maxClusters`	*Integer, default: 10*	*Max number of clusters.*
`restarts`	*Integer, default: 10*	*Number of restarts.*
`manual`	*Boolean, default: false*	*Manually select the number of clusters.*
`init`	*Boolean, default: false*	*Set whether to initialize using the probabilistic farthest first like method of the k-means++ algorithm (rather than the standard random selection of initial cluster centers).*
`distanceFunction`	*String, default: "Euclidean"*	*Distance function to use. Options are: Euclidean & Manhattan*

Argument	Type	Details
`maxIterations`	*Integer, default: null*	*Maximum number of iterations for k-means.*

ee.Clusterer.wekaCobweb

Implementation of the Cobweb clustering algorithm. For more information see:

D. Fisher (1987). Knowledge acquisition via incremental conceptual clustering. Machine Learning. 2(2):139-172. and J. H. Gennari, P. Langley, D. Fisher (1990). Models of incremental concept formation. Artificial Intelligence. 40:11-61.

Usage	Returns
`ee.Clusterer.wekaCobweb(`*`acuity`*`, `*`cutoff`*`, `*`seed`*`)`	Clusterer

Argument	Type	Details
`acuity`	*Float, default: 1*	*Acuity (minimum standard deviation).*
`cutoff`	*Float, default: 0.002*	*Cutoff (minimum category utility).*
`seed`	*Integer, default: 42*	*Random number seed.*

ee.Clusterer.wekaKMeans

Cluster data using the k means algorithm. Can use either the Euclidean distance (default) or the Manhattan distance. If the Manhattan distance is used, then centroids are computed as the component-wise median rather than mean. For more information see:

D. Arthur, S. Vassilvitskii: k-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 1027-1035, 2007.

Usage	Returns
`ee.Clusterer.wekaKMeans(nClusters, `*`init`*`, `*`canopies`*`, `*`maxCandidates`*`, `*`periodicPruning`*`, `*`minDensity`*`, `*`t1`*`, `*`t2`*`, `*`distanceFunction`*`, `*`maxIterations`*`, `*`preserveOrder`*`, `*`fast`*`, `*`seed`*`)`	Clusterer

Argument	Type	Details
`nClusters`	Integer	Number of clusters.

Argument	Type	Details
`init`	*Integer, default: 0*	*Initialization method to use.0 = random, 1 = k-means++, 2 = canopy, 3 = farthest first.*
`canopies`	*Boolean, default: false*	*Use canopies to reduce the number of distance calculations.*
`maxCandidates`	*Integer, default: 100*	*Maximum number of candidate canopies to retain in memory at any one time when using canopy clustering. T2 distance plus, data characteristics, will determine how many candidate canopies are formed before periodic and final pruning are performed, which might result in exceess memory consumption. This setting avoids large numbers of candidate canopies consuming memory.*
`periodicPruning`	*Integer, default: 10000*	*How often to prune low density canopies when using canopy clustering.*
`minDensity`	*Integer, default: 2*	*Minimum canopy density, when using canopy clustering, below which a canopy will be pruned during periodic pruning.*
`t1`	*Float, default: -1.5*	*The T1 distance to use when using canopy clustering. A value < 0 is taken as a positive multiplier for T2.*
`t2`	*Float, default: -1*	*The T2 distance to use when using canopy clustering. Values < 0 cause a heuristic based on attribute std. deviation to be used.*
`distanceFunction`	*String, default: "Euclidean"*	*Distance function to use. Options are: Euclidean & Manhattan*
`maxIterations`	*Integer, default: null*	*Maximum number of iterations.*
`preserveOrder`	*Boolean, default: false*	*Preserve order of instances.*
`fast`	*Boolean, default: false*	*Enables faster distance calculations, using cut-off values. Disables the calculation/output of squared errors/distances*
`seed`	*Integer, default: 10*	*The randomization seed.*

ee.Clusterer.wekaLVQ

A Clusterer that implements the Learning Vector Quantization algorithm. For more details, see:

T. Kohonen, "Learning Vector Quantization", The Handbook of Brain Theory and Neural Networks, 2nd Edition, MIT Press, 2003, pp. 631-634.

Usage	Returns
`ee.Clusterer.wekaLVQ(numClusters, learningRate, epochs, normalizeInput)`	Clusterer

Argument	Type	Details
`numClusters`	*Integer, default: 7*	*The number of clusters.*
`learningRate`	*Float, default: 1*	*The learning rate for the training algorithm. (Value should be greaterthan 0 and less or equal to 1).*
`epochs`	*Integer, default: 1000*	*Number of training epochs. (Value should be greater than or equal to 1).*
`normalizeInput`	*Boolean, default: false*	*Skip normalizing the attributes.*

ee.Clusterer.wekaXMeans

X-Means is K-Means with an efficient estimation of the number of clusters. For more information see:

Dan Pelleg, Andrew W. Moore: X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In: Seventeenth International Conference on Machine Learning, 727-734, 2000.

Usage	Returns
`ee.Clusterer.wekaXMeans(minClusters, maxClusters, maxIterations, maxKMeans, maxForChildren, useKD, cutoffFactor, distanceFunction, seed)`	Clusterer

Argument	Type	Details
`minClusters`	*Integer, default: 2*	*Minimum number of clusters.*
`maxClusters`	*Integer, default: 8*	*Maximum number of clusters.*
`maxIterations`	*Integer, default: 3*	*Maximum number of overall iterations.*
`maxKMeans`	*Integer, default: 1000*	*The maximum number of iterations to perform in KMeans.*
`maxForChildren`	*Integer, default: 1000*	*The maximum number of iterations in KMeans that is performed on the child centers.*
`useKD`	*Boolean, default: false*	*Use a KDTree.*
`cutoffFactor`	*Float, default: 0*	*Takes the given percentage of the split centroids if none of the children win.*
`distanceFunction`	*String, default: "Euclidean"*	*Distance function to use. Options are: Chebyshev, Euclidean & Manhattan.*

Argument	Type	Details
`seed`	*Integer, default: 10*	*The randomization seed.*

ee.ConfusionMatrix

Creates a confusion matrix. Axis 0 (the rows) of the matrix correspond to the actual values, and Axis 1 (the columns) to the predicted values.

Usage	Returns
`ee.ConfusionMatrix(array, order)`	ConfusionMatrix

Argument	Type	Details
`array`	Object	A square, 2D array of integers, representing the confusion matrix.
`order`	*List, default: null*	*The row and column size and order, for non-contiguous or non-zero based matrices.*

ee.ConfusionMatrix.accuracy

Computes the overall accuracy of a confusion matrix defined as correct / total.

Usage	Returns
`ConfusionMatrix.accuracy()`	Float

Argument	Type	Details
this: `confusionMatrix`	ConfusionMatrix	

ee.ConfusionMatrix.array

Returns a confusion matrix as an Array.

Usage	Returns
`ConfusionMatrix.array()`	Array

Argument	Type	Details
this: `confusionMatrix`	ConfusionMatrix	

ee.ConfusionMatrix.consumersAccuracy

Computes the consumer's accuracy (reliability) of a confusion matrix defined as (correct / total) for each row.

Usage	Returns
`ConfusionMatrix.consumersAccuracy()`	Array

Argument	Type	Details
this: `confusionMatrix`	ConfusionMatrix	

ee.ConfusionMatrix.fscore

Computes the Fβ-score for the confusion matrix.

Usage	Returns
`ConfusionMatrix.fscore(`*`beta`*`)`	Array

Argument	Type	Details
this: `confusionMatrix`	ConfusionMatrix	
`beta`	*Float, default: 1*	*A factor indicating how much more important recall is than precision. The standard F-score is equivalent to setting β to one.*

ee.ConfusionMatrix.kappa

Computes the Kappa statistic for the confusion matrix.

Usage	Returns
`ConfusionMatrix.kappa()`	Float

Argument	Type	Details
this: `confusionMatrix`	ConfusionMatrix	

ee.ConfusionMatrix.order

Returns the name and order of the rows and columns of the matrix.

Usage	Returns
`ConfusionMatrix.order()`	List

Argument	Type	Details
this: `confusionMatrix`	ConfusionMatrix	

ee.ConfusionMatrix.producersAccuracy

Computes the producer's accuracy of a confusion matrix defined as (correct / total) for each column.

Usage	Returns
`ConfusionMatrix.producersAccuracy()`	Array

Argument	Type	Details
this: `confusionMatrix`	ConfusionMatrix	

ee.Date

Constructs a new Date object.

Usage	Returns
`ee.Date(date, `*`tz`*`)`	Date

Argument	Type	Details
`date`	ComputedObject|Date|Number|String	The date to convert, one of: a number (number of milliseconds since the epoch), an ISO Date string, a JavaScript Date or a ComputedObject.
`tz`	*String, optional*	*An optional timezone only to be used with a string date.*

ee.Date.advance

Create a new Date by adding the specified units to the given Date.

Usage	Returns
`Date.advance(delta, unit, timeZone)`	Date

Argument	Type	Details
this: `date`	Date	
`delta`	Float	
`unit`	String	One of 'year', 'month' 'week', 'day', 'hour', 'minute', or 'second'.
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Date.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`Date.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Date.difference

Returns the difference between two Dates in the specified units; the result is floating-point and based on the average length of the unit.

Usage	Returns
`Date.difference(start, unit)`	Float

Argument	Type	Details
this: `date`	Date	
`start`	Date	
`unit`	String	One of 'year', 'month' 'week', 'day', 'hour', 'minute', or 'second'.

ee.Date.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Date.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Date.format

Convert a date to string.

Usage	Returns
`Date.format(`*`format`*`, `*`timeZone`*`)`	String

Argument	Type	Details
this: `date`	Date	
`format`	*String, default: null*	*A pattern, as described at http://joda-time.sourceforge.net/apidocs/org/joda/time/format/DateTimeFormat.html; if omitted will use ISO standard date formatting.*
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Date.fromYMD

Returns a Date given year, month, day.

Usage	Returns
`ee.Date.fromYMD(year, month, day, `*`timeZone`*`)`	Date

Argument	Type	Details
`year`	Integer	
`month`	Integer	
`day`	Integer	
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Date.get

Returns the specified unit of this date.

Usage	Returns
`Date.get(unit, timeZone)`	Long

Argument	Type	Details
this: `date`	Date	
`unit`	String	One of 'year', 'month' (returns 1-12), 'week' (1-53), 'day' (1-31), 'hour' (0-23), 'minute' (0-59), or 'second' (0-59).
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Date.getFraction

Returns this date's elapsed fraction of the specified unit (between 0 and 1).

Usage	Returns
`Date.getFraction(unit, timeZone)`	Float

Argument	Type	Details
this: `date`	Date	
`unit`	String	One of 'year', 'month' 'week', 'day', 'hour', 'minute', or 'second'.
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Date.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`Date.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Date.getRange

Returns a DateRange covering the unit of the specified type that contains this date, e.g. Date('2013-3-15').getRange('year') returns DateRange('2013-1-1', '2014-1-1').

Usage	Returns
`Date.getRange(unit,` *`timeZone`*`)`	DateRange

Argument	Type	Details
this: `date`	Date	
`unit`	String	One of 'year', 'month' 'week', 'day', 'hour', 'minute', or 'second'.
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Date.getRelative

Returns the specified (0-based) unit of this date relative to a larger unit, e.g. getRelative('day', 'year') returns a value between 0 and 365.

Usage	Returns
`Date.getRelative(unit, inUnit,` *`timeZone`*`)`	Long

Argument	Type	Details
this: `date`	Date	
`unit`	String	One of 'month' 'week', 'day', 'hour', 'minute', or 'second'.

Argument	Type	Details
`inUnit`	String	One of 'year', 'month' 'week', 'day', 'hour', or 'minute'.
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Date.millis

The number of milliseconds since 1970-01-01T00:00:00Z.

Usage	Returns
`Date.millis()`	Long

Argument	Type	Details
this: `date`	Date	

ee.Date.parse

Parse a date string, given a string describing its format.

Usage	Returns
`ee.Date.parse(format, date, `*`timeZone`*`)`	Date

Argument	Type	Details
`format`	String	A pattern, as described at http://joda-time.sourceforge.net/apidocs/org/joda/time/format/DateTimeFormat.html.
`date`	String	A string matching the given pattern.
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.Date.serialize

Returns the serialized representation of this object.

Usage	Returns
`Date.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Date.unitRatio

Returns the ratio of the length of one unit to the length of another, e.g. unitRatio('day', 'minute') returns 1440. Valid units are 'year', 'month' 'week', 'day', 'hour', 'minute', and 'second'.

Usage	Returns
`ee.Date.unitRatio(numerator, denominator)`	Float

Argument	Type	Details
`numerator`	String	
`denominator`	String	

ee.Date.update

Create a new Date by setting one or more of the units of the given Date to a new value. If a timeZone is given the new value(s) is interpreted in that zone.

Usage	Returns
`Date.update(`*`year`*`, `*`month`*`, `*`day`*`, `*`hour`*`, `*`minute`*`, `*`second`*`, `*`timeZone`*`)`	Date

Argument	Type	Details
this: `date`	Date	
`year`	*Integer, default: null*	
`month`	*Integer, default: null*	

Argument	Type	Details
`day`	*Integer, default: null*	
`hour`	*Integer, default: null*	
`minute`	*Integer, default: null*	
`second`	*Number, default: null*	
`timeZone`	*String, default: null*	*The time zone (e.g. 'America/Los_Angeles'); defaults to UTC.*

ee.DateRange

Creates a DateRange with the given start (inclusive) and end (exclusive), which may be Dates, numbers (interpreted as milliseconds since 1970-01-01T00:00:00Z), or strings (such as '1996-01-01T08:00'). If 'end' is not specified, a 1-millisecond range starting at 'start' is created.

Usage	Returns
`ee.DateRange(start, end, timeZone)`	DateRange

Argument	Type	Details
`start`	Object	
`end`	*Object, default: null*	
`timeZone`	*String, default: null*	*If start and/or end are provided as strings, the time zone in which to interpret them; defaults to UTC.*

ee.DateRange.contains

Returns true if the given Date or DateRange is within this DateRange.

Usage	Returns
`DateRange.contains(other)`	Boolean

Argument	Type	Details
this: `dateRange`	DateRange	
`other`	Object	

ee.DateRange.end

Returns the (exclusive) end of this DateRange.

Usage	Returns
`DateRange.end()`	Date

Argument	Type	Details
this: `dateRange`	DateRange	

ee.DateRange.intersection

Returns a DateRange that contains all points in the intersection of this DateRange and another.

Usage	Returns
`DateRange.intersection(other)`	DateRange

Argument	Type	Details
this: `dateRange`	DateRange	
`other`	DateRange	

ee.DateRange.intersects

Returns true if the given DateRange has at least one point in common with this DateRange.

Usage	Returns
`DateRange.intersects(other)`	Boolean

Argument	Type	Details
this: `dateRange`	DateRange	
`other`	DateRange	

ee.DateRange.isEmpty

Returns true if this DateRange contains no dates (i.e. start >= end).

Usage	Returns
`DateRange.isEmpty()`	Boolean

Argument	Type	Details
this: `dateRange`	DateRange	

ee.DateRange.isUnbounded

Returns true if this DateRange contains all dates.

Usage	Returns
`DateRange.isUnbounded()`	Boolean

Argument	Type	Details
this: `dateRange`	DateRange	

ee.DateRange.start

Returns the (inclusive) start of this DateRange.

Usage	Returns
`DateRange.start()`	Date

Argument	Type	Details
this: `dateRange`	DateRange	

ee.DateRange.unbounded

Returns a DateRange that includes all possible dates.

Usage	Returns
`ee.DateRange.unbounded()`	DateRange

No arguments.

ee.DateRange.union

Returns a DateRange that contains all points in the union of this DateRange and another.

Usage	Returns
`DateRange.union(other)`	DateRange

Argument	Type	Details
this: `dateRange`	DateRange	
`other`	DateRange	

ee.Dictionary

Constructs a new Dictionary.

Usage	Returns
`ee.Dictionary(`*`dict`*`)`	Dictionary

Argument	Type	Details
`dict`	*ComputedObject|Object, optional*	*An object to convert to a dictionary. This constructor accepts the following types: 1) Another dictionary. 2) A list of key/value pairs. 3) A null or no argument (producing an empty dictionary)*

ee.Dictionary.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

```
.filterBounds(geom).aside(print, 'In region')
.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')
.select('a', 'b');
```

Returns the same object, for chaining.

Usage	Returns
`Dictionary.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Dictionary.combine

Combines two dictionaries. In the case of duplicate names, the output will contain the value of the second dictionary unless overwrite is false. Null values in both dictionaries are ignored / removed.

Usage	Returns
`Dictionary.combine(second, overwrite)`	Dictionary

Argument	Type	Details
this: `first`	Dictionary	
`second`	Dictionary	
`overwrite`	*Boolean, default: true*	

ee.Dictionary.contains

Returns true if the dictionary contains the given key.

Usage	Returns
`Dictionary.contains(`*`key`*`)`	Boolean

Argument	Type	Details
this: `dictionary`	Dictionary	
`key`	*String, default: null*	

ee.Dictionary.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Dictionary.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Dictionary.fromLists

Construct a dictionary from two parallel lists of keys and values.

Usage	Returns
`ee.Dictionary.fromLists(keys, values)`	Dictionary

Argument	Type	Details
`keys`	List	
`values`	List	

ee.Dictionary.get

Extracts a named value from a dictionary. If the dictionary does not contain the given key, then defaultValue is returned, unless it is null.

Usage	Returns
`Dictionary.get(key, defaultValue)`	Object

Argument	Type	Details
this: `dictionary`	Dictionary	
`key`	String	
`defaultValue`	*Object, default: null*	

ee.Dictionary.getArray

Extracts a named array value from a dictionary.

Usage	Returns
`Dictionary.getArray(key)`	Array

Argument	Type	Details
this: `dictionary`	Dictionary	
`key`	String	

ee.Dictionary.getGeometry

Extracts a named geometry value from a dictionary.

Usage	Returns
`Dictionary.getGeometry(key)`	Geometry

Argument	Type	Details
this: `dictionary`	Dictionary	
`key`	String	

ee.Dictionary.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`Dictionary.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Dictionary.getNumber

Extracts a named number value from a dictionary.

Usage	Returns
`Dictionary.getNumber(key)`	Number

Argument	Type	Details
this: `dictionary`	Dictionary	
`key`	String	

ee.Dictionary.getString

Extracts a named string value from a dictionary.

Usage	Returns
`Dictionary.getString(key)`	String

Argument	Type	Details
this: `dictionary`	Dictionary	
`key`	String	

ee.Dictionary.keys

Retrieve the keys of a dictionary as a list. The keys will be sorted in natural order.

Usage	Returns
`Dictionary.keys()`	List

Argument	Type	Details
this: `dictionary`	Dictionary	

ee.Dictionary.map

Map an algorithm over a dictionary. The algorithm is expected to take 2 arguments, a key from the existing dictionary and the value it corresponds to, and return a new value for the given key. If the algorithm returns null, the key is dropped.

Usage	Returns
`Dictionary.map(baseAlgorithm)`	Dictionary

Argument	Type	Details
this: `dictionary`	Dictionary	
`baseAlgorithm`	Algorithm	

ee.Dictionary.remove

Returns a dictionary with the specified keys removed.

Usage	Returns
`Dictionary.remove(selectors, ignoreMissing)`	Dictionary

Argument	Type	Details
this: `dictionary`	Dictionary	
`selectors`	List	A list of keys names or regular expressions of key names to remove.
`ignoreMissing`	*Boolean, default: false*	*Ignore selectors that don't match at least 1 key.*

ee.Dictionary.rename

Rename elements in a dictionary.

Usage	Returns
`Dictionary.rename(from, to, overwrite)`	Dictionary

Argument	Type	Details
this: `dictionary`	Dictionary	
`from`	List	A list of keys to be renamed.
`to`	List	A list of the new names for the keys listed in the 'from' parameter. Must have the same length as the 'from' list.
`overwrite`	*Boolean, default: false*	*Allow overwriting existing properties with the same name.*

ee.Dictionary.select

Returns a dictionary with only the specified keys.

Usage	Returns
`Dictionary.select(selectors, ignoreMissing)`	Dictionary

Argument	Type	Details
this: `dictionary`	Dictionary	
`selectors`	List	A list of keys or regular expressions to select.
`ignoreMissing`	*Boolean, default: false*	*Ignore selectors that don't match at least 1 key.*

ee.Dictionary.serialize

Returns the serialized representation of this object.

Usage	Returns
`Dictionary.serialize(legacy)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Dictionary.set

Set a value in a dictionary.

Usage	Returns
`Dictionary.set(key, value)`	Dictionary

Argument	Type	Details
this: `dictionary`	Dictionary	
`key`	String	
`value`	Object	

ee.Dictionary.size

Returns the number of entries in a dictionary.

Usage	Returns
`Dictionary.size()`	Integer

Argument	Type	Details
this: `dictionary`	Dictionary	

ee.Dictionary.toArray

Returns numeric values of a dictionary as an array. If no keys are specified, all values are returned in the natural ordering of the dictionary's keys. The default 'axis' is 0.

Usage	Returns
`Dictionary.toArray(`*`keys`*`, `*`axis`*`)`	Array

Argument	Type	Details
this: `dictionary`	Dictionary	
`keys`	*List, default: null*	
`axis`	*Integer, default: 0*	

ee.Dictionary.toImage

Creates an image of constants from values in a dictionary. The bands of the image are ordered and named according to the names argument. If no names are specified, the bands are sorted alpha-numerically.

Usage	Returns
`Dictionary.toImage(`*`names`*`)`	Image

Argument	Type	Details
this: `dictionary`	Dictionary	The dictionary to convert.

Argument	Type	Details
`names`	*List, default: null*	*The order of the output bands.*

ee.Dictionary.values

Returns the values of a dictionary as a list. If no keys are specified, all values are returned in the natural ordering of the dictionary's keys.

Usage	Returns
`Dictionary.values(`*`keys`*`)`	List

Argument	Type	Details
this: `dictionary`	Dictionary	
`keys`	*List, default: null*	

ee.ErrorMargin

Returns an ErrorMargin of the given type with the given value.

Usage	Returns
`ee.ErrorMargin(`*`value`*`, `*`unit`*`)`	ErrorMargin

Argument	Type	Details
`value`	*Float, default: null*	*The maximum error value allowed by the margin. Ignored if the unit is 'infinite'.*
`unit`	*String, default: "meters"*	*The unit of this margin: 'meters', 'projected' or 'infinite'.*

ee.Feature

Features can be constructed from one of the following arguments plus an optional dictionary of properties:

- An ee.Geometry.

- A GeoJSON Geometry.

- A GeoJSON Feature.

- A computed object: reinterpreted as a geometry if properties are specified, and as a feature if they aren't.

Usage	Returns
`ee.Feature(geometry, properties)`	Feature

Argument	Type	Details
`geometry`	ComputedObject|Feature|Geometry|Object	A geometry or feature.
`properties`	*Object, optional*	*A dictionary of metadata properties. If the first parameter is a Feature (instead of a geometry), this is unused.*

ee.Feature.area

Returns the area of the feature's default geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`Feature.area(maxError, proj)`	Float

Argument	Type	Details
this: `feature`	Element	The feature from which the geometry is taken.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Feature.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

```
.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');
```

Returns the same object, for chaining.

Usage	Returns
`Feature.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Feature.bounds

Returns a feature containing the bounding box of the geometry of a given feature.

Usage	Returns
`Feature.bounds(`*`maxError`*`, `*`proj`*`)`	Feature

Argument	Type	Details
this: `feature`	Element	The feature the bound of which is being calculated.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Feature.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`Feature.buffer(distance, `*`maxError`*`, `*`proj`*`)`	Feature

Argument	Type	Details
this: `feature`	Element	The feature the geometry of which is being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Feature.centroid

Returns a feature containing the point at the center of the highest-dimension components of the geometry of a feature. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`Feature.centroid(`*`maxError`*`, `*`proj`*`)`	Feature

Argument	Type	Details
this: `feature`	Element	Calculates the centroid of this feature's default geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Feature.containedIn

Returns true if and only if the geometry of one feature is contained in the geometry of another.

Usage	Returns
`Feature.containedIn(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation.
`right`	Element	The feature containing the geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.contains

Returns true if and only if the geometry of one feature contains the geometry of another.

Usage	Returns
`Feature.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation.
`right`	Element	The feature containing the geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.convexHull

Returns the feature, with the geometry replaced by the convex hull of the original geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`Feature.convexHull(maxError, proj)`	Feature

Argument	Type	Details
this: `feature`	Element	The feature containing the geometry whole hull is being calculated.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.copyProperties

Copies metadata properties from one element to another.

Usage	Returns
`Feature.copyProperties(`*`source`*`, `*`properties`*`, `*`exclude`*`)`	Element

Argument	Type	Details
this: `destination`	*Element, default: null*	*The object whose properties to override.*
`source`	*Element, default: null*	*The object from which to copy the properties.*
`properties`	*List, default: null*	*The properties to copy. If omitted, all ordinary (i.e. non-system) properties are copied.*
`exclude`	*List, default: null*	*The list of properties to exclude when copying all properties. Must not be specified if properties is.*

ee.Feature.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`Feature.cutLines(distances, `*`maxError`*`, `*`proj`*`)`	Feature

Argument	Type	Details
this: `feature`	Element	Cuts the lines of this feature's default geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Feature.difference

Returns a feature with the properties of the 'left' feature, and the geometry that results from subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`Feature.difference(right, maxError, proj)`	Feature

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation. The properties of the result will be copied from this object.
`right`	Element	The feature containing the geometry used as the right operand of the operation. The properties of this object are ignored.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.disjoint

Returns true if and only if the feature geometries are disjoint.

Usage	Returns
`Feature.disjoint(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation.
`right`	Element	The feature containing the geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.dissolve

Returns a feature containing the union of the geometry of a feature. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`Feature.dissolve(`*`maxError`*`, `*`proj`*`)`	Element

Argument	Type	Details
this: `feature`	Element	The feature the geometry of which is being unioned.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Feature.distance

Returns the minimum distance between the geometries of two features.

Usage	Returns
`Feature.distance(right, `*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation.

Argument	Type	Details
`right`	Element	The feature containing the geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Feature.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Feature.geometry

Returns the geometry of a given feature in a given projection.

Usage	Returns
`Feature.geometry(`*`maxError`*`, `*`proj`*`, `*`geodesics`*`)`	Geometry

Argument	Type	Details
this: `feature`	Element	The feature from which the geometry is taken.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*

Argument	Type	Details
`proj`	*Projection, default: null*	*If specified, the geometry will be in this projection. If unspecified, the geometry will be in its default projection.*
`geodesics`	*Boolean, default: null*	*If true, the geometry will have geodesic edges. If false, it will have edges as straight lines in the specified projection. If null, the edge interpretation will be the same as the original geometry. This argument is ignored if proj is not specified.*

ee.Feature.get

Extract a property from a feature.

Usage	Returns
`Feature.get(property)`	

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.Feature.getArray

Extract a property from a feature.

Usage	Returns
`Feature.getArray(property)`	Array

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.Feature.getInfo

An imperative function that returns information about this feature via an AJAX call.

Returns a description of the feature.

Usage	Returns
`Feature.getInfo(`*`callback`*`)`	GeoJSONFeature

Argument	Type	Details
this: `feature`	Feature	The Feature instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. If supplied, will be called with the first parameter if successful and the second if unsuccessful.*

ee.Feature.getMapId

An imperative function that returns a map ID and optional token, suitable for generating a Map overlay.

Returns an object which may be passed to ee.data.getTileUrl or ui.Map.addLayer, including an additional 'image' field, containing a Collection.draw image wrapping a FeatureCollection containing this feature. Undefined if a callback was specified.

Usage	Returns
`Feature.getMapId(`*`visParams`*`, `*`callback`*`)`	MapId|Object

Argument	Type	Details
this: `feature`	Feature	The Feature instance.
`visParams`	*Object, optional*	*The visualization parameters. Currently only one parameter, 'color', containing an RGB color string is allowed. If visParams is not specified, black ("000000") is used.*
`callback`	*Function, optional*	*An async callback.*

ee.Feature.getNumber

Extract a property from a feature.

Usage	Returns
`Feature.getNumber(property)`	Number

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.Feature.getString

Extract a property from a feature.

Usage	Returns
`Feature.getString(property)`	String

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.Feature.hersDescriptor

Creates a dictionary of Histogram Error Ring Statistic (HERS) descriptor arrays from square array properties of an element. The HERS radius is taken to be the array's (side_length - 1) / 2.

Usage	Returns
`Feature.hersDescriptor(`*`selectors`*`, `*`buckets`*`, `*`peakWidthScale`*`)`	Dictionary

Argument	Type	Details
this: `element`	Element	The element with array properties.
`selectors`	*List, default: null*	*The array properties for which descriptors will be created. Selected array properties must be square, floating point arrays. Defaults to all array properties.*
`buckets`	*Integer, default: 100*	*The number of HERS buckets. Defaults to 100.*

Argument	Type	Details
`peakWidthScale`	*Float, default: 1*	*The HERS peak width scale. Defaults to 1.0.*

ee.Feature.id

Returns the ID of a given element within a collection. Objects outside collections are not guaranteed to have IDs.

Usage	Returns
`Feature.id()`	String

Argument	Type	Details
this: `element`	Element	The element from which the ID is taken.

ee.Feature.intersection

Returns a feature containing the intersection of the geometries of two features, with the properties of the left feature.

Usage	Returns
`Feature.intersection(right, `*`maxError`*`, `*`proj`*`)`	Feature

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation. The properties of the result will be copied from this object.
`right`	Element	The feature containing the geometry used as the right operand of the operation. The properties of this object are ignored.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.intersects

Returns true if and only if the feature geometries intersect.

Usage	Returns
`Feature.intersects(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation.
`right`	Element	The feature containing the geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.length

Returns the length of the linear parts of the geometry of a given feature. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`Feature.length(maxError, proj)`	Float

Argument	Type	Details
this: `feature`	Element	The feature from which the geometry is taken.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Feature.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry of a given feature. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`Feature.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `feature`	Element	The feature from which the geometry is taken.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Feature.propertyNames

Returns the names of properties on this element.

Usage	Returns
`Feature.propertyNames()`	List

Argument	Type	Details
this: `element`	Element	

ee.Feature.select

Selects properties from a feature by name or RE2-compatible regex and optionally renames them.

Usage	Returns
`Feature.select(propertySelectors, `*`newProperties`*`, `*`retainGeometry`*`)`	Element

Argument	Type	Details
this: `input`	Element	The feature to select properties from.
`propertySelectors`	List	A list of names or regexes specifying the properties to select.
`newProperties`	*List, default: null*	*Optional new names for the output properties. Must match the number of properties selected.*

Argument	Type	Details
`retainGeometry`	*Boolean, default: true*	*When false, the result will have a NULL geometry.*

ee.Feature.serialize

Returns the serialized representation of this object.

Usage	Returns
`Feature.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Feature.set

Overrides one or more metadata properties of an Element.

Returns the element with the specified properties overridden.

Usage	Returns
`Feature.set(var_args)`	Element

Argument	Type	Details
this: `element`	Element	The Element instance.
`var_args`	VarArgs	Either a dictionary of properties, or a vararg sequence of properties, e.g. key1, value1, key2, value2, ...

ee.Feature.setGeometry

Returns the feature, with the geometry replaced by the specified geometry.

Usage	Returns
`Feature.setGeometry(`*`geometry`*`)`	Element

Argument	Type	Details
this: `feature`	Element	The feature on which to set the geometry.
`geometry`	*Geometry, default: null*	*The geometry to set.*

ee.Feature.simplify

Simplifies the geometry of a feature to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`Feature.simplify(maxError, `*`proj`*`)`	Feature

Argument	Type	Details
this: `feature`	Element	The feature whose geometry is being simplified.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Feature.symmetricDifference

Returns a feature containing the symmetric difference between geometries of two features.

Usage	Returns
`Feature.symmetricDifference(right, `*`maxError`*`, `*`proj`*`)`	Feature

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation. The properties of the result will be copied from this object.
`right`	Element	The feature containing the geometry used as the right operand of the operation. The properties of this object are ignored.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.toArray

Creates an array from the given properties of an object, which must all be numbers.

Usage	Returns
`Feature.toArray(properties)`	Array

Argument	Type	Details
this: `feature`	Feature	The object from which to select array properties.
`properties`	List	The property selectors for each array element.

ee.Feature.toDictionary

Extract properties from a feature as a dictionary.

Usage	Returns
`Feature.toDictionary(`*`properties`*`)`	Dictionary

Argument	Type	Details
this: `element`	Element	The feature to extract the property from.
`properties`	*List, default: null*	*The list of properties to extract. Defaults to all non-system properties.*

ee.Feature.transform

Transforms the geometry of a feature to a specific projection.

Usage	Returns
`Feature.transform(`*`proj`*`, `*`maxError`*`)`	Feature

Argument	Type	Details
this: `feature`	Element	The feature the geometry of which is being converted.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Feature.union

Returns a feature containing the union of the geometries of two features.

Usage	Returns
`Feature.union(right, `*`maxError`*`, `*`proj`*`)`	Feature

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation. The properties of the result will be copied from this object.
`right`	Element	The feature containing the geometry used as the right operand of the operation. The properties of this object are ignored.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Feature.withinDistance

Returns true if and only if the geometries of two features are within a specified distance.

Usage	Returns
`Feature.withinDistance(right, distance, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Element	The feature containing the geometry used as the left operand of the operation.
`right`	Element	The feature containing the geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.FeatureCollection

FeatureCollections can be constructed from the following arguments:

- A string: assumed to be the name of a collection.

- A single geometry.

- A single feature.

- A list of features.

- A GeoJSON FeatureCollection

- A computed object: reinterpreted as a collection.

Usage	Returns
`ee.FeatureCollection(args, column)`	FeatureCollection

Argument	Type	Details
`args`	ComputedObject|Feature|FeatureCollection|Geometry|List	The constructor arguments.
`column`	*String, optional*	*The name of the geometry column to use. Only useful when working with a named collection.*

ee.FeatureCollection.aggregate_array

Aggregates over a given property of the objects in a collection, calculating a list of all the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_array(property)`	List

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_count

Aggregates over a given property of the objects in a collection, calculating the number of non-null values of the property.

Usage	Returns
`FeatureCollection.aggregate_count(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_count_distinct

Aggregates over a given property of the objects in a collection, calculating the number of distinct values for the selected property.

Usage	Returns
`FeatureCollection.aggregate_count_distinct(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_first

Aggregates over a given property of the objects in a collection, calculating the property value of the first object in the collection.

Usage	Returns
`FeatureCollection.aggregate_first(property)`	

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_histogram

Aggregates over a given property of the objects in a collection, calculating a histogram of the selected property.

Usage	Returns
`FeatureCollection.aggregate_histogram(property)`	Dictionary

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_max

Aggregates over a given property of the objects in a collection, calculating the maximum of the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_max(property)`	

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_mean

Aggregates over a given property of the objects in a collection, calculating the mean of the selected property.

Usage	Returns
`FeatureCollection.aggregate_mean(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_min

Aggregates over a given property of the objects in a collection, calculating the minimum of the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_min(property)`	

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_product

Aggregates over a given property of the objects in a collection, calculating the product of the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_product(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_sample_sd

Aggregates over a given property of the objects in a collection, calculating the sample std. deviation of the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_sample_sd(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_sample_var

Aggregates over a given property of the objects in a collection, calculating the sample variance of the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_sample_var(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_stats

Aggregates over a given property of the objects in a collection, calculating the sum, min, max, mean, sample standard deviation, sample variance, total standard deviation and total variance of the selected property.

Usage	Returns
`FeatureCollection.aggregate_stats(property)`	Dictionary

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_sum

Aggregates over a given property of the objects in a collection, calculating the sum of the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_sum(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_total_sd

Aggregates over a given property of the objects in a collection, calculating the total std. deviation of the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_total_sd(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aggregate_total_var

Aggregates over a given property of the objects in a collection, calculating the total variance of the values of the selected property.

Usage	Returns
`FeatureCollection.aggregate_total_var(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.FeatureCollection.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`FeatureCollection.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.FeatureCollection.classify

Classifies each feature in a collection.

Usage	Returns
`FeatureCollection.classify(classifier, outputName)`	FeatureCollection

Argument	Type	Details
this: `features`	FeatureCollection	The collection of features to classify. Each feature must contain all the properties in the classifier's schema.
`classifier`	Classifier	The classifier to use.
`outputName`	*String, default: "classification"*	*The name of the output property to be added. This argument is ignored if the classifier has more than one output.*

ee.FeatureCollection.cluster

Clusters each feature in a collection, adding a new column to each feature containing the cluster number to which it has been assigned.

Usage	Returns
`FeatureCollection.cluster(clusterer, outputName)`	FeatureCollection

Argument	Type	Details
this: `features`	FeatureCollection	The collection of features to cluster. Each feature must contain all the properties in the clusterer's schema.
`clusterer`	Clusterer	The clusterer to use.
`outputName`	*String, default: "cluster"*	*The name of the output property to be added.*

ee.FeatureCollection.copyProperties

Copies metadata properties from one element to another.

Usage	Returns
`FeatureCollection.copyProperties(`*`source`*`, `*`properties`*`, `*`exclude`*`)`	Element

Argument	Type	Details
this: `destination`	*Element, default: null*	*The object whose properties to override.*
`source`	*Element, default: null*	*The object from which to copy the properties.*
`properties`	*List, default: null*	*The properties to copy. If omitted, all ordinary (i.e. non-system) properties are copied.*
`exclude`	*List, default: null*	*The list of properties to exclude when copying all properties. Must not be specified if properties is.*

ee.FeatureCollection.distance

Produces a DOUBLE image where each pixel is the distance in meters from the pixel center to the nearest Point, LineString, or polygonal boundary in the collection. Note distance is also measured within interiors of polygons. Pixels that are not within 'searchRadius' meters of a geometry will be masked out.

Distances are computed on a sphere, so there is a small error proportional to the latitude difference between each pixel and the nearest geometry.

Usage	Returns
`FeatureCollection.distance(`*`searchRadius`*`, `*`maxError`*`)`	Image

Argument	Type	Details
this: `features`	FeatureCollection	Feature collection from which to get features used to compute pixel distances.
`searchRadius`	*Float, default: 100000*	*Maximum distance in meters from each pixel to look for edges. Pixels will be masked unless there are edges within this distance.*
`maxError`	*Float, default: 100*	*Maximum reprojection error in meters, only used if the input polylines require reprojection. If '0' is provided, then this operation will fail if projection is required.*

ee.FeatureCollection.distinct

Removes duplicates from a collection. Note that duplicates are determined using a strong hash over the serialized form of the selected properties.

Usage	Returns
`FeatureCollection.distinct(properties)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection from which objects will be selected.
`properties`	Object	A property name or a list of property names to use for comparison. The '.geo' property can be included to compare object geometries.

ee.FeatureCollection.draw

Paints a vector collection for visualization. Not intended for use as input to other algorithms.

Usage	Returns
`FeatureCollection.draw(color, pointRadius, strokeWidth)`	Image

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to draw.
`color`	String	A hex string in the format RRGGBB specifying the color to use for drawing the features.
`pointRadius`	*Integer, default: 3*	*The radius in pixels of the point markers.*
`strokeWidth`	*Integer, default: 2*	*The width in pixels of lines and polygon borders.*

ee.FeatureCollection.errorMatrix

Computes a 2D error matrix for a collection by comparing two columns of a collection: one containing the actual values, and one containing predicted values.The values are expected to be small contiguous integers, starting from 0. Axis 0 (the rows) of the matrix correspond to the actual values, and Axis 1 (the columns) to the predicted values.

Usage	Returns
`FeatureCollection.errorMatrix(actual, predicted, order)`	ConfusionMatrix

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection.
`actual`	String	The name of the property containing the actual value.
`predicted`	String	The name of the property containing the predicted value.
`order`	*List, default: null*	*A list of the expected values. If this argument is not specified, the values are assumed to be contiguous and span the range 0 to maxValue. If specified, only values matching this list are used, and the matrix will have dimensions and order matching the this list.*

ee.FeatureCollection.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`FeatureCollection.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.FeatureCollection.filter

Apply a filter to this collection.

Returns the filtered collection.

Usage	Returns
`FeatureCollection.filter(filter)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`filter`	Filter	A filter to apply to this collection.

ee.FeatureCollection.filterBounds

Shortcut to filter a collection by intersection with geometry. Items in the collection with a footprint that fails to intersect the given geometry will be excluded.

This is equivalent to this.filter(ee.Filter.bounds(...)).

Caution: providing a large or complex collection as the `geometry` argument can result in poor performance. Collating the geometry of collections does not scale well; use the smallest collection (or geometry) that is required to achieve the desired outcome.

Returns the filtered collection.

Usage	Returns
`FeatureCollection.filterBounds(geometry)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`geometry`	ComputedObject|FeatureCollection|Geometry	The geometry, feature or collection to intersect with.

ee.FeatureCollection.filterDate

Shortcut to filter a collection by a date range. The start and end may be Dates, numbers (interpreted as milliseconds since 1970-01-01T00:00:00Z), or strings (such as '1996-01-01T08:00'). Based on 'system:time_start'.

This is equivalent to this.filter(ee.Filter.date(...)); see the ee.Filter type for other date filtering options.

Returns the filtered collection.

Usage	Returns
`FeatureCollection.filterDate(start, end)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`start`	Date|Number|String	The start date (inclusive).
`end`	*Date|Number|String, optional*	*The end date (exclusive). Optional. If not specified, a 1-millisecond range starting at 'start' is created.*

ee.FeatureCollection.first

Returns the first entry from a given collection.

Usage	Returns
`FeatureCollection.first()`	Element

Argument	Type	Details
this: `collection`	FeatureCollection	The collection from which to select the first entry.

ee.FeatureCollection.flatten

Flattens collections of collections.

Usage	Returns
`FeatureCollection.flatten()`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection of collections.

ee.FeatureCollection.geometry

Extracts and merges the geometries of a collection. Requires that all the geometries in the collection share the projection and edge interpretation.

Caution: providing a large or complex collection as input can result in poor performance. Collating the geometry of collections does not scale well; use the smallest collection that is required to achieve the desired outcome.

Usage	Returns
`FeatureCollection.geometry(`*`maxError`*`)`	Geometry

Argument	Type	Details
this: `collection`	FeatureCollection	The collection whose geometries will be extracted.
`maxError`	*ErrorMargin, optional*	*An error margin to use when merging geometries.*

ee.FeatureCollection.get

Extract a property from a feature.

Usage	Returns
`FeatureCollection.get(property)`	

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.FeatureCollection.getArray

Extract a property from a feature.

Usage	Returns
`FeatureCollection.getArray(property)`	Array

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.FeatureCollection.getDownloadURL

Gets a download URL. When the URL is accessed, the FeatureCollection is downloaded in one of several formats.

Returns a download URL or undefined if a callback was specified.

Usage	Returns
`FeatureCollection.getDownloadURL(`*`format`*`, `*`selectors`*`, `*`filename`*`, `*`callback`*`)`	Object|String

Argument	Type	Details
this: `featurecollection`	FeatureCollection	The FeatureCollection instance.
`format`	*String, optional*	*The format of download, one of: "csv", "json", "geojson", "kml", "kmz" ("json" outputs GeoJSON). If unspecified, defaults to "csv".*
`selectors`	*List, optional*	*Feature property names used to select the attributes to be downloaded. If unspecified, all properties are included.*
`filename`	*String, optional*	*Name of the file to be downloaded; extension is appended by default. If unspecified, defaults to "table".*
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.FeatureCollection.getInfo

An imperative function that returns all the known information about this collection via an AJAX call.

Returns a collection description whose fields include:

- features: a list containing metadata about the features in the collection.

- properties: an optional dictionary containing the collection's metadata properties.

Usage	Returns
`FeatureCollection.getInfo(`*`callback`*`)`	FeatureCollectionDescription

Argument	Type	Details
this: `featurecollection`	FeatureCollection	The FeatureCollection instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. If supplied, will be called with the first parameter if successful and the second if unsuccessful.*

ee.FeatureCollection.getMapId

An imperative function that returns a map ID and optional token, suitable for generating a Map overlay.

Returns an object which may be passed to ee.data.getTileUrl or ui.Map.addLayer, including an additional 'image' field, containing a Collection.draw image wrapping a FeatureCollection containing this feature. Undefined if a callback was specified.

Usage	Returns
`FeatureCollection.getMapId(`*`visParams`*`, `*`callback`*`)`	MapId|Object

Argument	Type	Details
this: `featurecollection`	FeatureCollection	The FeatureCollection instance.
`visParams`	*Object, optional*	*The visualization parameters. Currently only one parameter, 'color', containing an RGB color string is allowed. If visParams is not specified, black ("000000") is used.*
`callback`	*Function, optional*	*An async callback.*

ee.FeatureCollection.getNumber

Extract a property from a feature.

Usage	Returns
`FeatureCollection.getNumber(property)`	Number

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.FeatureCollection.getString

Extract a property from a feature.

Usage	Returns
`FeatureCollection.getString(property)`	String

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.FeatureCollection.inverseDistance

Returns an inverse-distance weighted estimate of the value at each pixel.

Usage	Returns
`FeatureCollection.inverseDistance(range, propertyName, mean, stdDev, gamma, reducer)`	Image

Argument	Type	Details
this: `collection`	FeatureCollection	Feature collection to use as source data for the estimation.
`range`	Float	Size of the interpolation window (in meters).
`propertyName`	String	Name of the numeric property to be estimated.
`mean`	Float	Global expected mean.
`stdDev`	Float	Global standard deviation.
`gamma`	*Float, default: 1*	*Determines how quickly the estimates tend towards the global mean.*

Argument	Type	Details
`reducer`	*Reducer, default: null*	*Reducer used to collapse the 'propertyName' value of overlapping points into a single value.*

ee.FeatureCollection.iterate

Applies a user-supplied function to each element of a collection. The user-supplied function is given two arguments: the current element, and the value returned by the previous call to iterate() or the first argument, for the first iteration. The result is the value returned by the final call to the user-supplied function.

Returns the result of the Collection.iterate() call.

Usage	Returns
`FeatureCollection.iterate(algorithm, `*`first`*`)`	ComputedObject

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`algorithm`	Function	The function to apply to each element. Must take two arguments: an element of the collection and the value from the previous iteration.
`first`	*Object, optional*	*The initial state.*

ee.FeatureCollection.kriging

Returns the results of sampling a Kriging estimator at each pixel.

Usage	Returns
`FeatureCollection.kriging(propertyName, shape, range, sill, nugget, `*`maxDistance`*`, `*`reducer`*`)`	Image

Argument	Type	Details
this: `collection`	FeatureCollection	Feature collection to use as source data for the estimation.
`propertyName`	String	Property to be estimated (must be numeric).

Argument	Type	Details
`shape`	String	Semivariogram shape (one of {exponential, gaussian, spherical}).
`range`	Float	Semivariogram range, in meters.
`sill`	Float	Semivariogram sill.
`nugget`	Float	Semivariogram nugget.
`maxDistance`	*Float, default: null*	*Radius which determines which features are included in each pixel's computation, in meters. Defaults to the semivariogram's range.*
`reducer`	*Reducer, default: null*	*Reducer used to collapse the 'propertyName' value of overlapping points into a single value.*

ee.FeatureCollection.limit

Limit a collection to the specified number of elements, optionally sorting them by a specified property first.

Returns the limited collection.

Usage	Returns
`FeatureCollection.limit(max, `*`property`*`, `*`ascending`*`)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`max`	Number	The number to limit the collection to.
`property`	*String, optional*	*The property to sort by, if sorting.*
`ascending`	*Boolean, optional*	*Whether to sort in ascending or descending order. The default is true (ascending).*

ee.FeatureCollection.makeArray

Add a 1-D Array to each feature in a collection by combining a list of properties for each feature into a 1-D Array. All of the properties must be numeric values. If a feature doesn't contain all of the named properties, or any of them aren't numeric, the feature will be dropped from the resulting collection.

Usage	Returns
`FeatureCollection.makeArray(properties, `*`name`*`)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection from which properties will be selected.
`properties`	List	The properties to select.
`name`	*String, default: "array"*	*The name of the new array property.*

ee.FeatureCollection.map

Maps an algorithm over a collection.

Returns the mapped collection.

Usage	Returns
`FeatureCollection.map(algorithm, dropNulls)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`algorithm`	Function	The operation to map over the images or features of the collection. A JavaScript function that receives an image or features and returns one. The function is called only once and the result is captured as a description, so it cannot perform imperative operations or rely on external state.
`dropNulls`	*Boolean, optional*	*If true, the mapped algorithm is allowed to return nulls, and the elements for which it returns nulls will be dropped.*

ee.FeatureCollection.merge

Merges two collections into one. The result has all the elements that were in either collection.

Elements from the first collection will have IDs prefixed with "1*" and elements from the second collection will have IDs prefixed with "2*".

Note: If many collections need to be merged, consider placing them all in a collection and using FeatureCollection.flatten() instead. Repeated use of FeatureCollection.merge() will result in increasingly long element IDs and reduced performance.

Usage	Returns
`FeatureCollection.merge(collection2)`	FeatureCollection

Argument	Type	Details
this: `collection1`	FeatureCollection	The first collection to merge.
`collection2`	FeatureCollection	The second collection to merge.

ee.FeatureCollection.propertyNames

Returns the names of properties on this element.

Usage	Returns
`FeatureCollection.propertyNames()`	List

Argument	Type	Details
this: `element`	Element	

ee.FeatureCollection.randomColumn

Adds a column of deterministic pseudorandom numbers to a collection. The outputs are double-precision floating point numbers. When using the 'uniform' distribution (default), outputs are in the range of [0, 1). Using the 'normal' distribution, outputs have μ=0, **σ**=1, but have no explicit limits.

Usage	Returns
`FeatureCollection.randomColumn(`*`columnName`*`, `*`seed`*`, `*`distribution`*`)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection to which to add a random column.
`columnName`	*String, default: "random"*	*The name of the column to add.*
`seed`	*Long, default: 0*	*A seed used when generating the random numbers.*
`distribution`	*String, default: "uniform"*	*The distribution type of random numbers to produce; one of 'uniform' or 'normal'.*

ee.FeatureCollection.randomPoints

Generates points that are uniformly random in the given geometry. If the geometry is two-dimensional (polygon or multi-polygon) then the returned points are uniformly distributed on the given region of the sphere. If the geometry is one-dimensional (linestrings), the returned points are interpolated uniformly along the geometry's edges. If the geometry has dimension zero (points), the returned points are sampled uniformly from the input points. If a multi-geometry of mixed dimension is given, points are sampled from the component geometries with the highest dimension.

Usage	Returns
`ee.FeatureCollection.randomPoints(region, points, seed, maxError)`	FeatureCollection

Argument	Type	Details
`region`	Geometry	The region to generate points for.
`points`	*Integer, default: 1000*	*The number of points to generate.*
`seed`	*Long, default: 0*	*A seed for the random number generator.*
`maxError`	*ErrorMargin, optional*	*The maximum amount of error tolerated when performing any necessary reprojection.*

ee.FeatureCollection.reduceColumns

Apply a reducer to each element of a collection, using the given selectors to determine the inputs.

Returns a dictionary of results, keyed with the output names.

Usage	Returns
`FeatureCollection.reduceColumns(reducer, selectors, weightSelectors)`	Dictionary

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`reducer`	Reducer	The reducer to apply.
`selectors`	List	A selector for each input of the reducer.
`weightSelectors`	*List, default: null*	*A selector for each weighted input of the reducer.*

ee.FeatureCollection.reduceToImage

Creates an image from a feature collection by applying a reducer over the selected properties of all the features that intersect each pixel.

Usage	Returns
`FeatureCollection.reduceToImage(properties, reducer)`	Image

Argument	Type	Details
this: `collection`	FeatureCollection	Feature collection to intersect with each output pixel.
`properties`	List	Properties to select from each feature and pass into the reducer.
`reducer`	Reducer	A Reducer to combine the properties of each intersecting feature into a final result to store in the pixel.

ee.FeatureCollection.remap

Remaps the value of a specific property in a collection. Takes two parallel lists and maps values found in one to values in the other. Any element with a value that is not specified in the first list is dropped from the output collection.

Usage	Returns
`FeatureCollection.remap(lookupIn, lookupOut, columnName)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to be modified.
`lookupIn`	List	The input mapping values. Restricted to strings and integers.
`lookupOut`	List	The output mapping values. Must be the same size as lookupIn.
`columnName`	String	The name of the property to remap.

ee.FeatureCollection.select

Select properties from each Feature in a collection. It is also possible to call this function with only string arguments; they will be all be interpreted as propertySelectors (varargs).

Returns the feature collection with selected properties.

Usage	Returns
`FeatureCollection.select(propertySelectors, newProperties, retainGeometry)`	FeatureCollection

Argument	Type	Details
this: `featurecollection`	FeatureCollection	The FeatureCollection instance.
`propertySelectors`	List	A list of names or regexes specifying the attributes to select.
`newProperties`	*List, optional*	*A list of new names for the output properties. Must match the number of properties selected.*
`retainGeometry`	*Boolean, optional*	*When false, the result will have a NULL geometry. Defaults to true.*

ee.FeatureCollection.serialize

Returns the serialized representation of this object.

Usage	Returns
`FeatureCollection.serialize(legacy)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.FeatureCollection.set

Overrides one or more metadata properties of an Element.

Returns the element with the specified properties overridden.

Usage	Returns
`FeatureCollection.set(var_args)`	Element

Argument	Type	Details
this: `element`	Element	The Element instance.

Argument	Type	Details
`var_args`	VarArgs	Either a dictionary of properties, or a vararg sequence of properties, e.g. key1, value1, key2, value2, ...

ee.FeatureCollection.size

Returns the number of elements in the collection.

Usage	Returns
`FeatureCollection.size()`	Integer

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to count.

ee.FeatureCollection.sort

Sort a collection by the specified property.

Returns the sorted collection.

Usage	Returns
`FeatureCollection.sort(property, `*`ascending`*`)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`property`	String	The property to sort by.
`ascending`	*Boolean, optional*	*Whether to sort in ascending or descending order. The default is true (ascending).*

ee.FeatureCollection.style

Draw a vector collection for visualization using a simple style language.

Usage	Returns
`FeatureCollection.style(`*`color`*`, `*`pointSize`*`, `*`pointShape`*`, `*`width`*`, `*`fillColor`*`, `*`styleProperty`*`, `*`neighborhood`*`, `*`lineType`*`)`	Image

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to draw.
`color`	*String, default: "black"*	*A default color (CSS 3.0 color value e.g. 'FF0000' or 'red') to use for drawing the features. Supports opacity (e.g.: 'FF000088' for 50% transparent red).*
`pointSize`	*Integer, default: 3*	*The default size in pixels of the point markers.*
`pointShape`	*String, default: "circle"*	*The default shape of the marker to draw at each point location. One of: `circle`, `square`, `diamond`, `cross`, `plus`, `pentagram`, `hexagram`, `triangle`, `triangle_up`, `triangle_down`, `triangle_left`, `triangle_right`, `pentagon`, `hexagon`, `star5`, `star6`. This argument also supports the following Matlab marker abbreviations: `o`, `s`, `d`, `x`, `+`, `p`, `h`, `^`, `v`, `<`, `>`.*
`width`	*Float, default: 2*	*The default line width for lines and outlines for polygons and point shapes.*
`fillColor`	*String, default: null*	*The color for filling polygons and point shapes. Defaults to 'color' at 0.66 opacity.*
`styleProperty`	*String, default: null*	*A per-feature property expected to contain a dictionary. Values in the dictionary override any default values for that feature.*
`neighborhood`	*Integer, default: 5*	*If styleProperty is used and any feature has a pointSize or width larger than the defaults, tiling artifacts can occur. Specifies the maximum neighborhood (pointSize + width) needed for any feature.*
`lineType`	*String, default: "solid"*	*The default line style for lines and outlines of polygons and point shapes. Defaults to 'solid'. One of: solid, dotted, dashed.*

ee.FeatureCollection.toDictionary

Extract properties from a feature as a dictionary.

Usage	Returns
`FeatureCollection.toDictionary(`*`properties`*`)`	Dictionary

Argument	Type	Details
this: `element`	Element	The feature to extract the property from.
`properties`	*List, default: null*	*The list of properties to extract. Defaults to all non-system properties.*

ee.FeatureCollection.toList

Returns the elements of a collection as a list.

Usage	Returns
`FeatureCollection.toList(count, offset)`	List

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection to fetch.
`count`	Integer	The maximum number of elements to fetch.
`offset`	*Integer, default: 0*	*The number of elements to discard from the start. If set, (offset + count) elements will be fetched and the first offset elements will be discarded.*

ee.FeatureCollection.union

Merges all geometries in a given collection into one and returns a collection containing a single feature with only an ID of 'union_result' and a geometry.

Usage	Returns
`FeatureCollection.union(maxError)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The collection being merged.
`maxError`	*ErrorMargin, default: null*	*The maximum error allowed when performing any necessary reprojections. If not specified, defaults to the error margin requested from the output.*

ee.Filter

Constructs a new filter. This constructor accepts the following args:

- Another filter.

- A list of filters (which are implicitly ANDed together).

- A ComputedObject returning a filter. Users shouldn't be making these; they're produced by the generator functions below.

Usage	Returns
`ee.Filter(`*`filter`*`)`	Filter

Argument	Type	Details
`filter`	*Filter|List, optional*	*Optional filter to add.*

ee.Filter.and

Combine two or more filters using boolean AND.

Returns the constructed filter.

Usage	Returns
`ee.Filter.and(var_args)`	Filter

Argument	Type	Details
`var_args`	VarArgs	The filters to combine.

ee.Filter.area

Returns a filter that passes if the specified geometry has an area within the given range (inclusive).

Usage	Returns
`ee.Filter.area(min, max, `*`maxError`*`, `*`geometrySelector`*`)`	Filter

Argument	Type	Details
`min`	Float	Minimum area in square meters (inclusive).
`max`	Float	Maximum area in square meters (inclusive).
`maxError`	*ErrorMargin, default: null*	*The maximum allowed error for computing the geometry's area.*

Argument	Type	Details
`geometrySelector`	*String, default: null*	*The name of the geometry property to use for filtering. Leave blank or use '.geo' to operate on the object's geometry.*

ee.Filter.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`Filter.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Filter.bounds

Creates a filter that passes if the object's geometry intersects the given geometry.

Caution: providing a large or complex collection as the `geometry` argument can result in poor performance. Collating the geometry of collections does not scale well; use the smallest collection (or geometry) that is required to achieve the desired outcome.

Returns the constructed filter.

Usage	Returns
`ee.Filter.bounds(geometry, errorMargin)`	Filter

Argument	Type	Details
`geometry`	ComputedObject|FeatureCollection|Geometry	The geometry, feature or collection to intersect with.
`errorMargin`	*ComputedObject|Number, optional*	*An optional error margin. If a number, interpreted as sphere surface meters.*

ee.Filter.calendarRange

Returns a filter that passes if the object's timestamp falls within the given range of a calendar field. The `month`, `day_of_year`, `day_of_month`, and `day_of_week` are 1-based. Times are assumed to be in UTC. Weeks are assumed to begin on Monday as day 1. If `end` < `start` then this tests for `value` >= `start` OR `value` <= `end`, to allow for wrapping.

Usage	Returns
`ee.Filter.calendarRange(start, end, field)`	Filter

Argument	Type	Details
`start`	Integer	The start of the desired calendar field, inclusive.
`end`	*Integer, default: null*	*The end of the desired calendar field, inclusive. Defaults to the same value as start.*
`field`	*String, default: "day_of_year"*	*The calendar field to filter over. Options are: `year`, `month`, `hour`, `minute`, `day_of_year`, `day_of_month`, and `day_of_week`.*

ee.Filter.contains

Creates a unary or binary filter that passes if the left geometry contains the right geometry (empty geometries are not contained in anything).

Usage	Returns
`ee.Filter.contains(leftField, rightValue, rightField, leftValue, maxError)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*
`maxError`	*ErrorMargin, optional*	*The maximum reprojection error allowed during filter application.*

ee.Filter.date

Filter a collection by date range. The start and end may be Dates, numbers

(interpreted as milliseconds since 1970-01-01T00:00:00Z), or strings (such as '1996-01-01T08:00'). Based on 'system:time_start' property.

Returns the constructed filter.

Usage	Returns
`ee.Filter.date(start, end)`	Filter

Argument	Type	Details
`start`	Date|Number|String	The start date (inclusive).
`end`	*Date|Number|String, optional*	*The end date (exclusive). Optional. If not specified, a 1-millisecond range starting at 'start' is created.*

ee.Filter.dateRangeContains

Creates a unary or binary filter that passes if the left operand, a date range, contains the right operand, a date.

Usage	Returns
`ee.Filter.dateRangeContains(leftField, rightValue, rightField, leftValue)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*

Argument	Type	Details
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.dayOfYear

Returns a filter that passes if the object's timestamp falls within the given day-of-year range.

Usage	Returns
`ee.Filter.dayOfYear(start, end)`	Filter

Argument	Type	Details
`start`	Integer	The start of the desired day range, inclusive.
`end`	Integer	The end of the desired day range, inclusive.

ee.Filter.disjoint

Creates a unary or binary filter that passes unless the left geometry intersects the right geometry.

Usage	Returns
`ee.Filter.disjoint(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`, `*`maxError`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*
`maxError`	*ErrorMargin, optional*	*The maximum reprojection error allowed during filter application.*

ee.Filter.eq

Filter to metadata equal to the given value.

Returns the constructed filter.

Usage	Returns
`ee.Filter.eq(name, value)`	Filter

Argument	Type	Details
`name`	String	The property name to filter on.
`value`	Object	The value to compare against.

ee.Filter.equals

Creates a unary or binary filter that passes if the two operands are equals.

Usage	Returns
`ee.Filter.equals(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Filter.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Filter.expression

Constructs a filter tree from a string.

Usage	Returns
`ee.Filter.expression(expression)`	Filter

Argument	Type	Details
`expression`	String	A string to be parsed into a Filter object (e.g.: "property > value"). Supported operators include: >, >=, <, <=, ==, !=, (), !, && and ||.

ee.Filter.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`Filter.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.

Argument	Type	Details
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Filter.greaterThan

Creates a unary or binary filter that passes if the left operand is greater than the right operand.

Usage	Returns
`ee.Filter.greaterThan(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.greaterThanOrEquals

Creates a unary or binary filter that passes unless the left operand is less than the right operand.

Usage	Returns
`ee.Filter.greaterThanOrEquals(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.gt

Filter on metadata greater than the given value.

Returns the constructed filter.

Usage	Returns
`ee.Filter.gt(name, value)`	Filter

Argument	Type	Details
`name`	String	The property name to filter on.
`value`	Object	The value to compare against.

ee.Filter.gte

Filter on metadata greater than or equal to the given value.

Returns the constructed filter.

Usage	Returns
`ee.Filter.gte(name, value)`	Filter

Argument	Type	Details
`name`	String	The property name to filter on.
`value`	Object	The value to compare against.

ee.Filter.hasType

Creates a unary or binary filter that passes if the left operand has the type, or is a subtype of the type named in the right operand.

Usage	Returns
`ee.Filter.hasType(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*

Argument	Type	Details
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.inList

Filter on metadata contained in a list.

Returns the constructed filter.

Usage	Returns
`ee.Filter.inList(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, optional*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*List, optional*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, optional*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*List, optional*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.intersects

Creates a unary or binary filter that passes if the left geometry intersects the right geometry.

Usage	Returns
`ee.Filter.intersects(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`, `*`maxError`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*

Argument	Type	Details
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*
`maxError`	*ErrorMargin, optional*	*The maximum reprojection error allowed during filter application.*

ee.Filter.isContained

Creates a unary or binary filter that passes if the right geometry contains the left geometry (empty geometries are not contained in anything).

Usage	Returns
`ee.Filter.isContained(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`, `*`maxError`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*
`maxError`	*ErrorMargin, optional*	*The maximum reprojection error allowed during filter application.*

ee.Filter.lessThan

Creates a unary or binary filter that passes if the left operand is less than the right operand.

Usage	Returns
`ee.Filter.lessThan(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.lessThanOrEquals

Creates a unary or binary filter that passes unless the left operand is greater than the right operand.

Usage	Returns
`ee.Filter.lessThanOrEquals(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.listContains

Creates a unary or binary filter that passes if the left operand, a list, contains the right operand.

Usage	Returns
`ee.Filter.listContains(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.lt

Filter to metadata less than the given value.

Returns the constructed filter.

Usage	Returns
`ee.Filter.lt(name, value)`	Filter

Argument	Type	Details
`name`	String	The property name to filter on.
`value`	Object	The value to compare against.

ee.Filter.lte

Filter on metadata less than or equal to the given value.

Returns the constructed filter.

Usage	Returns
`ee.Filter.lte(name, value)`	Filter

Argument	Type	Details
`name`	String	The property name to filter on.
`value`	Object	The value to compare against.

ee.Filter.maxDifference

Creates a unary or binary filter that passes if the left and right operands, both numbers, are within a given maximum difference. If used as a join condition, this numeric difference is used as a join measure.

Usage	Returns
`ee.Filter.maxDifference(difference, leftField, rightValue, rightField, leftValue)`	Filter

Argument	Type	Details
`difference`	Float	The maximum difference for which the filter will return true.
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*

Argument	Type	Details
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.neq

Filter to metadata not equal to the given value.

Returns the constructed filter.

Usage	Returns
`ee.Filter.neq(name, value)`	Filter

Argument	Type	Details
`name`	String	The property name to filter on.
`value`	Object	The value to compare against.

ee.Filter.not

Returns the opposite of the input filter, i.e. the resulting filter will match if and only if the input filter doesn't match.

Usage	Returns
`Filter.not()`	Filter

Argument	Type	Details
this: `filter`	Filter	The Filter instance.

ee.Filter.notEquals

Creates a unary or binary filter that passes unless the two operands are equals.

Usage	Returns
`ee.Filter.notEquals(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.notNull

Returns a filter that passes if all the named properties are not null.

Usage	Returns
`ee.Filter.notNull(properties)`	Filter

Argument	Type	Details
`properties`	List	

ee.Filter.or

Combine two or more filters using boolean OR.

Returns the constructed filter.

Usage	Returns
`ee.Filter.or(var_args)`	Filter

Argument	Type	Details
`var_args`	VarArgs	The filters to combine.

ee.Filter.rangeContains

Returns a filter that passes if the value of the selected field is in the specified range (inclusive).

Usage	Returns
`ee.Filter.rangeContains(field, minValue, maxValue)`	Filter

Argument	Type	Details
`field`	String	A selector for the property being tested.
`minValue`	Object	The lower bound of the range.
`maxValue`	Object	The upper bound of the range.

ee.Filter.serialize

Returns the serialized representation of this object.

Usage	Returns
`Filter.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Filter.stringContains

Creates a unary or binary filter that passes if the left operand, a string, contains the right operand, also a string.

Usage	Returns
`ee.Filter.stringContains(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*

Argument	Type	Details
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.stringEndsWith

Creates a unary or binary filter that passes if the left operand, a string, ends with the right operand, also a string.

Usage	Returns
`ee.Filter.stringEndsWith(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.stringStartsWith

Creates a unary or binary filter that passes if the left operand, a string, starts with the right operand, also a string.

Usage	Returns
`ee.Filter.stringStartsWith(`*`leftField`*`, `*`rightValue`*`, `*`rightField`*`, `*`leftValue`*`)`	Filter

Argument	Type	Details
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*

ee.Filter.withinDistance

Creates a unary or binary filter that passes if the left geometry is within a specified distance of the right geometry. If used as a join condition, this distance is used as a join measure.

Usage	Returns
`ee.Filter.withinDistance(distance, leftField, rightValue, rightField, leftValue, maxError)`	Filter

Argument	Type	Details
`distance`	Float	The maximum distance for which the filter will return true.
`leftField`	*String, default: null*	*A selector for the left operand. Should not be specified if leftValue is specified.*
`rightValue`	*Object, default: null*	*The value of the right operand. Should not be specified if rightField is specified.*
`rightField`	*String, default: null*	*A selector for the right operand. Should not be specified if rightValue is specified.*
`leftValue`	*Object, default: null*	*The value of the left operand. Should not be specified if leftField is specified.*
`maxError`	*ErrorMargin, optional*	*The maximum reprojection error allowed during filter application.*

ee.Geometry

Creates a geometry.

Usage	Returns
`ee.Geometry(geoJson, proj, geodesic, evenOdd)`	Geometry

Argument	Type	Details
`geoJson`	Object	The GeoJSON object describing the geometry or a ComputedObject to be reinterpreted as a Geometry. Supports CRS specifications as per the GeoJSON spec, but only allows named (rather than "linked" CRSs). If this includes a 'geodesic' field, and opt_geodesic is not specified, it will be used as opt_geodesic.
`proj`	*Projection, optional*	*An optional projection specification, either as a CRS ID code or as a WKT string. If specified, overrides any CRS found in the geoJson parameter. If unspecified and the geoJson does not declare a CRS, defaults to "EPSG:4326" (x=longitude, y=latitude).*
`geodesic`	*Boolean, optional*	*Whether line segments should be interpreted as spherical geodesics. If false, indicates that line segments should be interpreted as planar lines in the specified CRS. If absent, defaults to true if the CRS is geographic (including the default EPSG:4326), or to false if the CRS is projected.*

Argument	Type	Details
evenOdd	*Boolean, optional*	*If true, polygon interiors will be determined by the even/odd rule, where a point is inside if it crosses an odd number of edges to reach a point at infinity. Otherwise polygons use the left- inside rule, where interiors are on the left side of the shell's edges when walking the vertices in the given order. If unspecified, defaults to true.*

ee.Geometry.BBox

Constructs a rectangle whose edges are lines of latitude and longitude.

The result is a planar WGS84 rectangle.

If (east - west) ≥ 360 then the longitude range will be normalized to -180 to

+180; otherwise they will be treated as designating points on a circle (e.g. east may be numerically less than west).

Usage	Returns
`ee.Geometry.BBox(west, south, east, north)`	Geometry.BBox

Argument	Type	Details
west	Number	The westernmost enclosed longitude. Will be adjusted to lie in the range -180° to 180°.
south	Number	The southernmost enclosed latitude. If less than -90° (south pole), will be treated as -90°.
east	Number	The easternmost enclosed longitude.
north	Number	The northernmost enclosed latitude. If greater than +90° (north pole), will be treated as +90°.

ee.Geometry.BBox.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
BBox.area(*maxError*, *proj*)	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.BBox.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`BBox.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.BBox.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`BBox.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.BBox.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`BBox.buffer(distance, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.BBox.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`BBox.centroid(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.BBox.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`BBox.containedIn(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`BBox.contains(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.

Argument	Type	Details
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`BBox.convexHull(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`BBox.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.BBox.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`BBox.coveringGrid(proj, scale)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.BBox.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`BBox.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.BBox.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`BBox.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`BBox.disjoint(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`BBox.dissolve(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.BBox.distance

Returns the minimum distance between two geometries.

Usage	Returns
`BBox.distance(right, `*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`BBox.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.BBox.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`BBox.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.BBox.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`BBox.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.BBox.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`BBox.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.BBox.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`BBox.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.BBox.intersection

Returns the intersection of the two geometries.

Usage	Returns
`BBox.intersection(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`BBox.intersects(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`BBox.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.BBox.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`BBox.length(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.BBox.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`BBox.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.BBox.projection

Returns the projection of the geometry.

Usage	Returns
`BBox.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.BBox.serialize

Returns the serialized representation of this object.

Usage	Returns
`BBox.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.BBox.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`BBox.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.

Argument	Type	Details
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.BBox.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`BBox.symmetricDifference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`BBox.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.BBox.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`BBox.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.BBox.transform

Transforms the geometry to a specific projection.

Usage	Returns
`BBox.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.BBox.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`BBox.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.BBox.union

Returns the union of the two geometries.

Usage	Returns
`BBox.union(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.BBox.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`BBox.withinDistance(right, distance, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString

Constructs an ee.Geometry describing a LineString.

For convenience, varargs may be used when all arguments are numbers. This allows creating geodesic EPSG:4326 LineStrings given an even number of arguments, e.g. ee.Geometry.LineString(aLng, aLat, bLng, bLat, ...).

Usage	Returns
`ee.Geometry.LineString(coords, proj, geodesic, maxError)`	Geometry.LineString

Argument	Type	Details
`coords`	List	A list of at least two points. May be a list of coordinates in the GeoJSON 'LineString' format, a list of at least two ee.Geometry objects describing a point, or a list of at least four numbers defining the [x,y] coordinates of at least two points.
`proj`	*Projection, optional*	*The projection of this geometry. If unspecified, the default is the projection of the input ee.Geometry, or EPSG:4326 if there are no ee.Geometry inputs.*
`geodesic`	*Boolean, optional*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, optional*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*

ee.Geometry.LineString.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`LineString.area(maxError, proj)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.LineString.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`LineString.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.LineString.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`LineString.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.LineString.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`LineString.buffer(distance, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.LineString.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`LineString.centroid(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.LineString.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`LineString.containedIn(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`LineString.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`LineString.convexHull(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`LineString.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LineString.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`LineString.coveringGrid(proj, `*`scale`*`)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is

Argument	Type	Details
		scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.LineString.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`LineString.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.LineString.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`LineString.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`LineString.disjoint(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`LineString.dissolve(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*

Argument	Type	Details
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.LineString.distance

Returns the minimum distance between two geometries.

Usage	Returns
`LineString.distance(right, maxError, proj)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`LineString.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LineString.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`LineString.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.LineString.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`LineString.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LineString.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`LineString.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LineString.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`LineString.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.LineString.intersection

Returns the intersection of the two geometries.

Usage	Returns
`LineString.intersection(right,` *`maxError`*`,` *`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`LineString.intersects(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`LineString.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LineString.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`LineString.length(maxError, proj)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.LineString.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`LineString.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.LineString.projection

Returns the projection of the geometry.

Usage	Returns
`LineString.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LineString.serialize

Returns the serialized representation of this object.

Usage	Returns
`LineString.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.LineString.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`LineString.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.LineString.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`LineString.symmetricDifference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`LineString.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.LineString.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`LineString.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.LineString.transform

Transforms the geometry to a specific projection.

Usage	Returns
`LineString.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.LineString.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`LineString.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LineString.union

Returns the union of the two geometries.

Usage	Returns
`LineString.union(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LineString.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`LineString.withinDistance(right, distance, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing

Constructs an ee.Geometry describing a LinearRing. If the last point is not equal to the first, a duplicate of the first point will be added at the end.

For convenience, varargs may be used when all arguments are numbers. This allows creating geodesic EPSG:4326 LinearRings given an even number of arguments, e.g. ee.Geometry.LinearRing(aLng, aLat, bLng, bLat, ..., aLng, aLat).

Usage	Returns
`ee.Geometry.LinearRing(coords, proj, geodesic, maxError)`	Geometry.LinearRing

Argument	Type	Details
`coords`	List	A list of points in the ring. May be a list of coordinates in the GeoJSON 'LinearRing' format, a list of at least three ee.Geometry objects describing a point, or a list of at least six numbers defining the [x,y] coordinates of at least three points.
`proj`	*Projection, optional*	*The projection of this geometry. If unspecified, the default is the projection of the input ee.Geometry, or EPSG:4326 if there are no ee.Geometry inputs.*
`geodesic`	*Boolean, optional*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, optional*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*

ee.Geometry.LinearRing.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`LinearRing.area(maxError, proj)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.LinearRing.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

```
.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')
.filterBounds(geom).aside(print, 'In region')
.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')
.select('a', 'b');
```

Returns the same object, for chaining.

Usage	Returns
`LinearRing.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.LinearRing.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`LinearRing.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.LinearRing.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`LinearRing.buffer(distance, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.LinearRing.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`LinearRing.centroid(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.LinearRing.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`LinearRing.containedIn(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`LinearRing.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`LinearRing.convexHull(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`LinearRing.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LinearRing.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`LinearRing.coveringGrid(proj, `*`scale`*`)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is

Argument	Type	Details
		scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.LinearRing.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`LinearRing.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.LinearRing.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`LinearRing.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`LinearRing.disjoint(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`LinearRing.dissolve(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*

Argument	Type	Details
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.LinearRing.distance

Returns the minimum distance between two geometries.

Usage	Returns
`LinearRing.distance(right, maxError, proj)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`LinearRing.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LinearRing.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`LinearRing.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.LinearRing.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`LinearRing.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LinearRing.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`LinearRing.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LinearRing.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`LinearRing.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.LinearRing.intersection

Returns the intersection of the two geometries.

Usage	Returns
`LinearRing.intersection(right,` *`maxError`*`,` *`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`LinearRing.intersects(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`LinearRing.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LinearRing.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`LinearRing.length(maxError, proj)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.LinearRing.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`LinearRing.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.LinearRing.projection

Returns the projection of the geometry.

Usage	Returns
`LinearRing.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LinearRing.serialize

Returns the serialized representation of this object.

Usage	Returns
`LinearRing.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.LinearRing.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`LinearRing.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.LinearRing.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`LinearRing.symmetricDifference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`LinearRing.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.LinearRing.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`LinearRing.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.LinearRing.transform

Transforms the geometry to a specific projection.

Usage	Returns
`LinearRing.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.LinearRing.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`LinearRing.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.LinearRing.union

Returns the union of the two geometries.

Usage	Returns
`LinearRing.union(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.LinearRing.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`LinearRing.withinDistance(right, distance, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString

Constructs an ee.Geometry describing a MultiLineString.

For convenience, varargs may be used when all arguments are numbers. This allows creating geodesic EPSG:4326 MultiLineStrings with a single LineString, given an even number of arguments, e.g. ee.Geometry.MultiLineString(aLng, aLat, bLng, bLat, ...).

Usage	Returns
`ee.Geometry.MultiLineString(coords, proj, geodesic, maxError)`	Geometry.MultiLineString

Argument	Type	Details
`coords`	List	A list of linestrings. May be a list of coordinates in the GeoJSON 'MultiLineString' format, a list of at least two ee.Geometry objects describing a LineString, or a list of numbers defining a single linestring.
`proj`	*Projection, optional*	*The projection of this geometry. If unspecified, the default is the projection of the input ee.Geometry, or EPSG:4326 if there are no ee.Geometry inputs.*
`geodesic`	*Boolean, optional*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, optional*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*

ee.Geometry.MultiLineString.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`MultiLineString.area(maxError, proj)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.MultiLineString.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

```
.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')
.filterBounds(geom).aside(print, 'In region')
.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')
.select('a', 'b');
```

Returns the same object, for chaining.

Usage	Returns
`MultiLineString.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.MultiLineString.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`MultiLineString.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.MultiLineString.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`MultiLineString.buffer(distance, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.MultiLineString.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`MultiLineString.centroid(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.MultiLineString.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`MultiLineString.containedIn(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`MultiLineString.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`MultiLineString.convexHull(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`MultiLineString.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiLineString.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`MultiLineString.coveringGrid(proj, `*`scale`*`)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is

Argument	Type	Details
		scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.MultiLineString.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`MultiLineString.cutLines(distances, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.MultiLineString.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`MultiLineString.difference(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`MultiLineString.disjoint(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`MultiLineString.dissolve(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*

Argument	Type	Details
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.MultiLineString.distance

Returns the minimum distance between two geometries.

Usage	Returns
`MultiLineString.distance(right, maxError, proj)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`MultiLineString.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiLineString.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`MultiLineString.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.MultiLineString.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`MultiLineString.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiLineString.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`MultiLineString.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiLineString.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`MultiLineString.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.MultiLineString.intersection

Returns the intersection of the two geometries.

Usage	Returns
`MultiLineString.intersection(right,` *`maxError`*`,` *`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`MultiLineString.intersects(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`MultiLineString.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiLineString.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`MultiLineString.length(maxError, proj)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.MultiLineString.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`MultiLineString.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.MultiLineString.projection

Returns the projection of the geometry.

Usage	Returns
`MultiLineString.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiLineString.serialize

Returns the serialized representation of this object.

Usage	Returns
`MultiLineString.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.MultiLineString.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`MultiLineString.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.MultiLineString.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`MultiLineString.symmetricDifference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`MultiLineString.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.MultiLineString.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`MultiLineString.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.MultiLineString.transform

Transforms the geometry to a specific projection.

Usage	Returns
`MultiLineString.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.MultiLineString.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`MultiLineString.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiLineString.union

Returns the union of the two geometries.

Usage	Returns
`MultiLineString.union(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiLineString.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`MultiLineString.withinDistance(right, distance, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint

Constructs an ee.Geometry describing a MultiPoint.

For convenience, varargs may be used when all arguments are numbers. This allows creating EPSG:4326 MultiPoints given an even number of arguments, e.g. ee.Geometry.MultiPoint(aLng, aLat, bLng, bLat, ...).

Usage	Returns
`ee.Geometry.MultiPoint(coords, `*`proj`*`)`	Geometry.MultiPoint

Argument	Type	Details
`coords`	List	A list of points, each in the GeoJSON 'coordinates' format of a Point, or a list of the x,y coordinates in the given projection, or an ee.Geometry describing a point.
`proj`	*Projection, optional*	*The projection of this geometry. If unspecified, the default is the projection of the input ee.Geometry, or EPSG:4326 if there are no ee.Geometry inputs.*

ee.Geometry.MultiPoint.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`MultiPoint.area(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.MultiPoint.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`MultiPoint.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.MultiPoint.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`MultiPoint.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.MultiPoint.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`MultiPoint.buffer(distance, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.

Argument	Type	Details
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.MultiPoint.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`MultiPoint.centroid(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.MultiPoint.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`MultiPoint.containedIn(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`MultiPoint.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`MultiPoint.convexHull(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`MultiPoint.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPoint.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`MultiPoint.coveringGrid(proj, `*`scale`*`)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.MultiPoint.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`MultiPoint.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.MultiPoint.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`MultiPoint.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`MultiPoint.disjoint(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`MultiPoint.dissolve(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.MultiPoint.distance

Returns the minimum distance between two geometries.

Usage	Returns
`MultiPoint.distance(right, maxError, proj)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`MultiPoint.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPoint.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`MultiPoint.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.

Argument	Type	Details
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.MultiPoint.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`MultiPoint.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPoint.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`MultiPoint.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPoint.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over

getInfo().

Returns the computed value of this object.

Usage	Returns
`MultiPoint.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.MultiPoint.intersection

Returns the intersection of the two geometries.

Usage	Returns
`MultiPoint.intersection(right,` *`maxError`*`,` *`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`MultiPoint.intersects(right,` *`maxError`*`,` *`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`MultiPoint.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPoint.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`MultiPoint.length(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.MultiPoint.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`MultiPoint.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.MultiPoint.projection

Returns the projection of the geometry.

Usage	Returns
`MultiPoint.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPoint.serialize

Returns the serialized representation of this object.

Usage	Returns
`MultiPoint.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.MultiPoint.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`MultiPoint.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.MultiPoint.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`MultiPoint.symmetricDifference(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.

Argument	Type	Details
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`MultiPoint.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.MultiPoint.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`MultiPoint.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.MultiPoint.transform

Transforms the geometry to a specific projection.

Usage	Returns
`MultiPoint.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.MultiPoint.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`MultiPoint.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPoint.union

Returns the union of the two geometries.

Usage	Returns
`MultiPoint.union(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPoint.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`MultiPoint.withinDistance(right, distance, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon

Constructs an ee.Geometry describing a MultiPolygon.

For convenience, varargs may be used when all arguments are numbers. This allows creating geodesic EPSG:4326 MultiPolygons with a single Polygon with a single LinearRing given an even number of arguments, e.g. ee.Geometry.MultiPolygon(aLng, aLat, bLng, bLat, ..., aLng, aLat).

Usage	Returns
`ee.Geometry.MultiPolygon(coords, proj, geodesic, maxError, evenOdd)`	Geometry.MultiPolygon

Argument	Type	Details
`coords`	List	A list of polygons. May be a list of coordinates in the GeoJSON 'MultiPolygon' format, a list of ee.Geometry objects describing a Polygon, or a list of numbers defining a single polygon boundary.
`proj`	*Projection, optional*	*The projection of this geometry. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, optional*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, optional*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*
`evenOdd`	*Boolean, optional*	*If true, polygon interiors will be determined by the even/odd rule, where a point is inside if it crosses an odd number of edges to reach a point at infinity. Otherwise polygons use the left- inside rule, where interiors are on the left side of the shell's edges when walking the vertices in the given order. If unspecified, defaults to true.*

ee.Geometry.MultiPolygon.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`MultiPolygon.area(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.MultiPolygon.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

```
.filterBounds(geom).aside(print, 'In region')
.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')
.select('a', 'b');
```

Returns the same object, for chaining.

Usage	Returns
`MultiPolygon.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.MultiPolygon.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`MultiPolygon.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.MultiPolygon.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`MultiPolygon.buffer(distance, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.MultiPolygon.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`MultiPolygon.centroid(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.MultiPolygon.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`MultiPolygon.containedIn(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`MultiPolygon.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`MultiPolygon.convexHull(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`MultiPolygon.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPolygon.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`MultiPolygon.coveringGrid(proj, `*`scale`*`)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is

Argument	Type	Details
		scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.MultiPolygon.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`MultiPolygon.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.MultiPolygon.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`MultiPolygon.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`MultiPolygon.disjoint(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`MultiPolygon.dissolve(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*

Argument	Type	Details
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.MultiPolygon.distance

Returns the minimum distance between two geometries.

Usage	Returns
`MultiPolygon.distance(right, maxError, proj)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`MultiPolygon.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPolygon.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`MultiPolygon.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.MultiPolygon.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`MultiPolygon.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPolygon.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`MultiPolygon.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPolygon.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`MultiPolygon.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.MultiPolygon.intersection

Returns the intersection of the two geometries.

Usage	Returns
`MultiPolygon.intersection(right,` *`maxError`*`,` *`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`MultiPolygon.intersects(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`MultiPolygon.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPolygon.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`MultiPolygon.length(maxError, proj)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.MultiPolygon.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`MultiPolygon.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.MultiPolygon.projection

Returns the projection of the geometry.

Usage	Returns
`MultiPolygon.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPolygon.serialize

Returns the serialized representation of this object.

Usage	Returns
`MultiPolygon.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.MultiPolygon.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`MultiPolygon.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.MultiPolygon.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`MultiPolygon.symmetricDifference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`MultiPolygon.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.MultiPolygon.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`MultiPolygon.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.MultiPolygon.transform

Transforms the geometry to a specific projection.

Usage	Returns
`MultiPolygon.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.MultiPolygon.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`MultiPolygon.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.MultiPolygon.union

Returns the union of the two geometries.

Usage	Returns
`MultiPolygon.union(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.MultiPolygon.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`MultiPolygon.withinDistance(right, distance, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point

Constructs an ee.Geometry describing a point.

For convenience, varargs may be used when all arguments are numbers. This allows creating EPSG:4326 points, e.g. ee.Geometry.Point(lng, lat).

Usage	Returns
`ee.Geometry.Point(coords, `*`proj`*`)`	Geometry.Point

Argument	Type	Details
`coords`	List	A list of two [x,y] coordinates in the given projection.
`proj`	*Projection, optional*	*The projection of this geometry, or EPSG:4326 if unspecified.*

ee.Geometry.Point.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`Point.area(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.Point.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`Point.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.Point.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`Point.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.Point.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`Point.buffer(distance, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.

Argument	Type	Details
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.Point.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`Point.centroid(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.Point.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`Point.containedIn(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`Point.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`Point.convexHull(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`Point.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Point.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`Point.coveringGrid(proj, `*`scale`*`)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.Point.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`Point.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.Point.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`Point.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`Point.disjoint(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`Point.dissolve(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.Point.distance

Returns the minimum distance between two geometries.

Usage	Returns
`Point.distance(right, maxError, proj)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`Point.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Point.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Point.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.

Argument	Type	Details
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.Point.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`Point.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Point.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`Point.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Point.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over

getInfo().

Returns the computed value of this object.

Usage	Returns
`Point.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.Point.intersection

Returns the intersection of the two geometries.

Usage	Returns
`Point.intersection(right,` *`maxError`*`,` *`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`Point.intersects(right,` *`maxError`*`,` *`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`Point.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Point.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`Point.length(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.Point.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`Point.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.Point.projection

Returns the projection of the geometry.

Usage	Returns
`Point.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Point.serialize

Returns the serialized representation of this object.

Usage	Returns
`Point.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.Point.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`Point.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.Point.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`Point.symmetricDifference(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.

Argument	Type	Details
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`Point.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.Point.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`Point.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.Point.transform

Transforms the geometry to a specific projection.

Usage	Returns
`Point.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.Point.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`Point.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Point.union

Returns the union of the two geometries.

Usage	Returns
`Point.union(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Point.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`Point.withinDistance(right, distance, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon

Constructs an ee.Geometry describing a polygon.

For convenience, varargs may be used when all arguments are numbers. This allows creating geodesic EPSG:4326 Polygons with a single LinearRing given an even number of arguments, e.g. ee.Geometry.Polygon(aLng, aLat, bLng, bLat, ..., aLng, aLat).

Usage	Returns
`ee.Geometry.Polygon(coords, `*`proj`*`, `*`geodesic`*`, `*`maxError`*`, `*`evenOdd`*`)`	Geometry.Polygon

Argument	Type	Details
`coords`	List	A list of rings defining the boundaries of the polygon. May be a list of coordinates in the GeoJSON 'Polygon' format, a list of ee.Geometry objects describing a LinearRing, or a list of numbers defining a single polygon boundary.
`proj`	*Projection, optional*	*The projection of this geometry. The default is the projection of the inputs, where Numbers are assumed to be EPSG:4326.*
`geodesic`	*Boolean, optional*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`maxError`	*ErrorMargin, optional*	*Max error when input geometry must be reprojected to an explicitly requested result projection or geodesic state.*
`evenOdd`	*Boolean, optional*	*If true, polygon interiors will be determined by the even/odd rule, where a point is inside if it crosses an odd number of edges to reach a point at infinity. Otherwise polygons use the left- inside rule, where interiors are on the left side of the shell's edges when walking the vertices in the given order. If unspecified, defaults to true.*

ee.Geometry.Polygon.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`Polygon.area(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.Polygon.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

```
.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')
.filterBounds(geom).aside(print, 'In region')
.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')
.select('a', 'b');
```

Returns the same object, for chaining.

Usage	Returns
`Polygon.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.Polygon.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`Polygon.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.Polygon.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`Polygon.buffer(distance, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.Polygon.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`Polygon.centroid(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.Polygon.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`Polygon.containedIn(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`Polygon.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`Polygon.convexHull(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`Polygon.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Polygon.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`Polygon.coveringGrid(proj, `*`scale`*`)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is

Argument	Type	Details
		scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.Polygon.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`Polygon.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.Polygon.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`Polygon.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`Polygon.disjoint(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`Polygon.dissolve(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*

Argument	Type	Details
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.Polygon.distance

Returns the minimum distance between two geometries.

Usage	Returns
`Polygon.distance(right, maxError, proj)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`Polygon.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Polygon.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Polygon.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.Polygon.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`Polygon.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Polygon.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`Polygon.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Polygon.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`Polygon.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.Polygon.intersection

Returns the intersection of the two geometries.

Usage	Returns
`Polygon.intersection(right,` *`maxError`*`,` *`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`Polygon.intersects(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`Polygon.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Polygon.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`Polygon.length(maxError, proj)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.Polygon.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`Polygon.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.Polygon.projection

Returns the projection of the geometry.

Usage	Returns
`Polygon.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Polygon.serialize

Returns the serialized representation of this object.

Usage	Returns
`Polygon.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.Polygon.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`Polygon.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.Polygon.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`Polygon.symmetricDifference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`Polygon.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.Polygon.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`Polygon.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.Polygon.transform

Transforms the geometry to a specific projection.

Usage	Returns
`Polygon.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.Polygon.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`Polygon.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Polygon.union

Returns the union of the two geometries.

Usage	Returns
`Polygon.union(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Polygon.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`Polygon.withinDistance(right, distance, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle

Constructs an ee.Geometry describing a rectangular polygon.

For convenience, varargs may be used when all arguments are numbers. This allows creating EPSG:4326 Polygons given exactly four coordinates, e.g. ee.Geometry.Rectangle(minLng, minLat, maxLng, maxLat).

Usage	Returns
`ee.Geometry.Rectangle(coords, proj, geodesic, evenOdd)`	Geometry.Rectangle

Argument	Type	Details
`coords`	List	The minimum and maximum corners of the rectangle, as a list of two points each in the format of GeoJSON 'Point' coordinates, or a list of two ee.Geometry objects describing a point, or a list of four numbers in the order xMin, yMin, xMax, yMax.
`proj`	*Projection, optional*	*The projection of this geometry. If unspecified, the default is the projection of the input ee.Geometry, or EPSG:4326 if there are no ee.Geometry inputs.*
`geodesic`	*Boolean, optional*	*If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth. The default is the geodesic state of the inputs, or true if the inputs are numbers.*
`evenOdd`	*Boolean, optional*	*If true, polygon interiors will be determined by the even/odd rule, where a point is inside if it crosses an odd number of edges to reach a point at infinity. Otherwise polygons use the left- inside rule, where interiors are on the left side of the shell's edges when walking the vertices in the given order. If unspecified, defaults to true.*

ee.Geometry.Rectangle.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`Rectangle.area(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.Rectangle.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

```
.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')
.select('a', 'b');
```

Returns the same object, for chaining.

Usage	Returns
`Rectangle.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.Rectangle.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`Rectangle.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.Rectangle.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`Rectangle.buffer(distance, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.Rectangle.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`Rectangle.centroid(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.Rectangle.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`Rectangle.containedIn(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`Rectangle.contains(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`Rectangle.convexHull(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`Rectangle.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Rectangle.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`Rectangle.coveringGrid(proj, scale)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.Rectangle.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`Rectangle.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.Rectangle.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`Rectangle.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`Rectangle.disjoint(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`Rectangle.dissolve(maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.Rectangle.distance

Returns the minimum distance between two geometries.

Usage	Returns
`Rectangle.distance(right, maxError, proj)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`Rectangle.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Rectangle.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Rectangle.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.

Argument	Type	Details
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.Rectangle.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`Rectangle.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Rectangle.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`Rectangle.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Rectangle.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over

getInfo().

Returns the computed value of this object.

Usage	Returns
`Rectangle.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.Rectangle.intersection

Returns the intersection of the two geometries.

Usage	Returns
`Rectangle.intersection(right,` *`maxError`*`,` *`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`Rectangle.intersects(right,` *`maxError`*`,` *`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`Rectangle.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Rectangle.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`Rectangle.length(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.Rectangle.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`Rectangle.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.Rectangle.projection

Returns the projection of the geometry.

Usage	Returns
`Rectangle.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Rectangle.serialize

Returns the serialized representation of this object.

Usage	Returns
`Rectangle.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.Rectangle.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`Rectangle.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.Rectangle.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`Rectangle.symmetricDifference(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.

Argument	Type	Details
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`Rectangle.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.Rectangle.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`Rectangle.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.Rectangle.transform

Transforms the geometry to a specific projection.

Usage	Returns
`Rectangle.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.Rectangle.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`Rectangle.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.Rectangle.union

Returns the union of the two geometries.

Usage	Returns
`Rectangle.union(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.Rectangle.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`Rectangle.withinDistance(right, distance, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.area

Returns the area of the geometry. Area of points and line strings is 0, and the area of multi geometries is the sum of the areas of their components (intersecting areas are counted multiple times).

Usage	Returns
`Geometry.area(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The geometry input.

Argument	Type	Details
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in square meters.*

ee.Geometry.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`Geometry.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Geometry.bounds

Returns the bounding rectangle of the geometry.

Usage	Returns
`Geometry.bounds(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Return the bounding box of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.buffer

Returns the input buffered by a given distance. If the distance is positive, the geometry is expanded, and if the distance is negative, the geometry is contracted.

Usage	Returns
`Geometry.buffer(distance, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry being buffered.
`distance`	Float	The distance of the buffering, which may be negative. If no projection is specified, the unit is meters. Otherwise the unit is in the coordinate system of the projection.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when approximating the buffering circle and performing any necessary reprojection. If unspecified, defaults to 1% of the distance.*
`proj`	*Projection, default: null*	*If specified, the buffering will be performed in this projection and the distance will be interpreted as units of the coordinate system of this projection. Otherwise the distance is interpereted as meters and the buffering is performed in a spherical coordinate system.*

ee.Geometry.centroid

Returns a point at the center of the highest-dimension components of the geometry. Lower-dimensional components are ignored, so the centroid of a geometry containing two polygons, three lines and a point is equivalent to the centroid of a geometry containing just the two polygons.

Usage	Returns
`Geometry.centroid(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the centroid of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in WGS84.*

ee.Geometry.containedIn

Returns true if and only if one geometry is contained in the other.

Usage	Returns
`Geometry.containedIn(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.contains

Returns true if and only if one geometry contains the other.

Usage	Returns
`Geometry.contains(right, `*`maxError`*`, `*`proj`*`)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*

Argument	Type	Details
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.convexHull

Returns the convex hull of the given geometry. The convex hull of a single point is the point itself, the convex hull of collinear points is a line, and the convex hull of everything else is a polygon. Note that a degenerate polygon with all vertices on the same line will result in a line segment.

Usage	Returns
`Geometry.convexHull(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Calculates the convex hull of this geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.coordinates

Returns a GeoJSON-style list of the geometry's coordinates.

Usage	Returns
`Geometry.coordinates()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.coveringGrid

Returns a collection of features that cover this geometry, where each feature is a rectangle in the grid defined by the given projection.

Usage	Returns
`Geometry.coveringGrid(proj, scale)`	FeatureCollection

Argument	Type	Details
this: `geometry`	Geometry	The result is the grid cells that intersect with this region.
`proj`	Projection	The projection in which to construct the grid. A feature is generated for each grid cell that intersects 'geometry', where cell corners are at integer-valued positions in the projection. If the projection is scaled in meters, the points will be on a grid of that size at the point of true scale.
`scale`	*Float, default: null*	*Overrides the scale of the projection, if provided. May be required if the projection isn't already scaled.*

ee.Geometry.cutLines

Converts LineString, MultiLineString, and LinearRing geometries into a MultiLineString by cutting them into parts no longer than the given distance along their length. All other geometry types will be converted to an empty MultiLineString.

Usage	Returns
`Geometry.cutLines(distances, maxError, proj)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	Cuts the lines of this geometry.
`distances`	List	Distances along each LineString to cut the line into separate pieces, measured in units of the given proj, or meters if proj is unspecified.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*Projection of the result and distance measurements, or WGS84 if unspecified.*

ee.Geometry.difference

Returns the result of subtracting the 'right' geometry from the 'left' geometry.

Usage	Returns
`Geometry.difference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.disjoint

Returns true if and only if the geometries are disjoint.

Usage	Returns
`Geometry.disjoint(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.dissolve

Returns the union of the geometry. This leaves single geometries untouched, and unions multi geometries.

Usage	Returns
`Geometry.dissolve(`*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to union.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the union will be performed in this projection. Otherwise it will be performed in a spherical coordinate system.*

ee.Geometry.distance

Returns the minimum distance between two geometries.

Usage	Returns
`Geometry.distance(right, `*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.edgesAreGeodesics

Returns true if the geometry edges, if any, are geodesics along a spherical model of the earth; if false, any edges are straight lines in the projection.

Usage	Returns
`Geometry.edgesAreGeodesics()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Geometry.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Geometry.geodesic

If false, edges are straight in the projection. If true, edges are curved to follow the shortest path on the surface of the Earth.

Usage	Returns
`Geometry.geodesic()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.geometries

Returns the list of geometries in a GeometryCollection, or a singleton list of the geometry for single geometries.

Usage	Returns
`Geometry.geometries()`	List

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`Geometry.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Geometry.intersection

Returns the intersection of the two geometries.

Usage	Returns
`Geometry.intersection(right, `*`maxError`*`, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.intersects

Returns true if and only if the geometries intersect.

Usage	Returns
`Geometry.intersects(right, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.isUnbounded

Returns whether the geometry is unbounded.

Usage	Returns
`Geometry.isUnbounded()`	Boolean

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.length

Returns the length of the linear parts of the geometry. Polygonal parts are ignored. The length of multi geometries is the sum of the lengths of their components.

Usage	Returns
`Geometry.length(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.perimeter

Returns the length of the perimeter of the polygonal parts of the geometry. The perimeter of multi geometries is the sum of the perimeters of their components.

Usage	Returns
`Geometry.perimeter(`*`maxError`*`, `*`proj`*`)`	Float

Argument	Type	Details
this: `geometry`	Geometry	The input geometry.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the result will be in the units of the coordinate system of this projection. Otherwise it will be in meters.*

ee.Geometry.projection

Returns the projection of the geometry.

Usage	Returns
`Geometry.projection()`	Projection

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.serialize

Returns the serialized representation of this object.

Usage	Returns
`Geometry.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Geometry.simplify

Simplifies the geometry to within a given error margin. Note that this does not respect the error margin requested by the consumer of this algorithm, unless maxError is explicitly specified to be null.

This overrides the default Earth Engine policy for propagating error margins, so regardless of the geometry accuracy requested from the output, the inputs will be requested with the error margin specified in the arguments to this algorithm. This results in consistent rendering at all zoom levels of a rendered vector map, but at lower zoom levels (i.e. zoomed out), the geometry won't be simplified, which may harm performance.

Usage	Returns
`Geometry.simplify(maxError, `*`proj`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to simplify.

Argument	Type	Details
`maxError`	ErrorMargin	The maximum amount of error by which the result may differ from the input.
`proj`	*Projection, default: null*	*If specified, the result will be in this projection. Otherwise it will be in the same projection as the input. If the error margin is in projected units, the margin will be interpreted as units of this projection*

ee.Geometry.symmetricDifference

Returns the symmetric difference between two geometries.

Usage	Returns
`Geometry.symmetricDifference(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.toGeoJSON

Returns a GeoJSON representation of the geometry.

Usage	Returns
`Geometry.toGeoJSON()`	GeoJSONGeometry

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.toGeoJSONString

Returns a GeoJSON string representation of the geometry.

Usage	Returns
`Geometry.toGeoJSONString()`	String

Argument	Type	Details
this: `geometry`	Geometry	The Geometry instance.

ee.Geometry.transform

Transforms the geometry to a specific projection.

Usage	Returns
`Geometry.transform(`*`proj`*`, `*`maxError`*`)`	Geometry

Argument	Type	Details
this: `geometry`	Geometry	The geometry to reproject.
`proj`	*Projection, optional*	*The target projection. Defaults to WGS84. If this has a geographic CRS, the edges of the geometry will be interpreted as geodesics. Otherwise they will be interpreted as straight lines in the projection.*
`maxError`	*ErrorMargin, default: null*	*The maximum projection error.*

ee.Geometry.type

Returns the GeoJSON type of the geometry.

Usage	Returns
`Geometry.type()`	String

Argument	Type	Details
this: `geometry`	Geometry	

ee.Geometry.union

Returns the union of the two geometries.

Usage	Returns
`Geometry.union(right, maxError, proj)`	Geometry

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Geometry.withinDistance

Returns true if and only if the geometries are within a specified distance.

Usage	Returns
`Geometry.withinDistance(right, distance, maxError, proj)`	Boolean

Argument	Type	Details
this: `left`	Geometry	The geometry used as the left operand of the operation.
`right`	Geometry	The geometry used as the right operand of the operation.
`distance`	Float	The distance threshold. If a projection is specified, the distance is in units of that projected coordinate system, otherwise it is in meters.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*The projection in which to perform the operation. If not specified, the operation will be performed in a spherical coordinate system, and linear distances will be in meters on the sphere.*

ee.Image

An object to represent an Earth Engine image. This constructor accepts a variety of arguments:

- A string: an EarthEngine asset id,

- A string and a number: an EarthEngine asset id and version,

- A number or ee.Array: creates a constant image,

- A list: creates an image out of each list element and combines them into a single image,

- An ee.Image: returns the argument,

- Nothing: results in an empty transparent image.

Usage	Returns
`ee.Image(`*`args`*`)`	Image

Argument	Type	Details
`args`	*Image|List, optional*	*Constructor argument.*

ee.Image.abs

Computes the absolute value of the input.

Usage	Returns
`Image.abs()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.acos

Computes the arc cosine in radians of the input.

Usage	Returns
`Image.acos()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.add

Adds the first value to the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.add(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.addBands

Returns an image containing all bands copied from the first input and selected bands from the second input, optionally overwriting bands in the first image with the same name. The new image has the metadata and footprint from the first input image.

Usage	Returns
`Image.addBands(srcImg, `*`names`*`, `*`overwrite`*`)`	Image

Argument	Type	Details
this: `dstImg`	Image	An image into which to copy bands.
`srcImg`	Image	An image containing bands to copy.
`names`	*List, default: null*	*Optional list of band names to copy. If names is omitted, all bands from srcImg will be copied over.*

Argument	Type	Details
`overwrite`	*Boolean, default: false*	*If true, bands from `srcImg` will override bands with the same names in `dstImg`. Otherwise the new band will be renamed with a numerical suffix (`foo` to `foo_1` unless `foo_1` exists, then `foo_2` unless it exists, etc).*

ee.Image.and

Returns 1 if and only if both values are non-zero for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is boolean.

Usage	Returns
`Image.and(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.arrayAccum

Accumulates elements of each array pixel along the given axis, by setting each element of the result array pixel to the reduction of elements in that pixel along the given axis, up to and including the current position on the axis. May be used to make a cumulative sum, a monotonically increasing sequence, etc.

Usage	Returns
`Image.arrayAccum(axis, `*`reducer`*`)`	Image

Argument	Type	Details
this: `input`	Image	Input image.
`axis`	Integer	Axis along which to perform the cumulative sum.

Argument	Type	Details
`reducer`	*Reducer, default: null*	*Reducer to accumulate values. Default is SUM, to produce the cumulative sum of each vector along the given axis.*

ee.Image.arrayArgmax

Computes the positional indices of the maximum value in image of array values. If there are multiple occurrences of the maximum, the indices reflect the first.

Usage	Returns
`Image.arrayArgmax()`	Image

Argument	Type	Details
this: `image`	Image	The input image.

ee.Image.arrayCat

Creates an array image by concatenating each array pixel along the given axis in each band.

Usage	Returns
`Image.arrayCat(image2, axis)`	Image

Argument	Type	Details
this: `image1`	Image	First array image to concatenate.
`image2`	Image	Second array image to concatenate.
`axis`	Integer	Axis to concatenate along.

ee.Image.arrayDimensions

Returns the number of dimensions in each array band, and 0 for scalar image bands.

Usage	Returns
`Image.arrayDimensions()`	Image

Argument	Type	Details
this: `input`	Image	Input image.

ee.Image.arrayDotProduct

Computes the dot product of each pair of 1-D arrays in the bands of the input images.

Usage	Returns
`Image.arrayDotProduct(image2)`	Image

Argument	Type	Details
this: `image1`	Image	First array image of 1-D vectors.
`image2`	Image	Second array image of 1-D vectors.

ee.Image.arrayFlatten

Converts a single band image of equal-shape multidimensional pixels to an image of scalar pixels, with one band for each element of the array.

Usage	Returns
`Image.arrayFlatten(coordinateLabels, `*`separator`*`)`	Image

Argument	Type	Details
this: `image`	Image	Image of multidimensional pixels to flatten.
`coordinateLabels`	List	Name of each position along each axis. For example, 2x2 arrays with axes meaning 'day' and 'color' could have labels like [['monday', 'tuesday'], ['red', 'green']], resulting in band names'monday_red', 'monday_green', 'tuesday_red', and 'tuesday_green'.
`separator`	*String, default: "_"*	*Separator between array labels in each band name.*

ee.Image.arrayGet

For each band, an output band of the same name is created with the value at the given position extracted from the input multidimensional pixel in that band.

Usage	Returns
`Image.arrayGet(position)`	Image

Argument	Type	Details
this: `image`	Image	Array to get an element from.
`position`	Image	The coordinates of the element to get. There must be as many scalar bands as there are dimensions in the input image.

ee.Image.arrayLength

Returns the length of each pixel's array along the given axis.

Usage	Returns
`Image.arrayLength(axis)`	Image

Argument	Type	Details
this: `input`	Image	Input image.
`axis`	Integer	The axis along which to take the length.

ee.Image.arrayLengths

Returns a 1D array image with the length of each array axis.

Usage	Returns
`Image.arrayLengths()`	Image

Argument	Type	Details
this: `input`	Image	Input image.

ee.Image.arrayMask

Creates an array image where each array-valued pixel is masked with another array-valued pixel, retaining only the elements where the mask is non-zero. If the mask image has one band it will be applied to all the bands of 'input', otherwise they must have the same number of bands.

Usage	Returns
`Image.arrayMask(mask)`	Image

Argument	Type	Details
this: `input`	Image	Array image to mask.
`mask`	Image	Array image to mask with.

ee.Image.arrayPad

Pads the array values in each pixel to be a fixed length. The pad value will be appended to each array to extend it to given length along each axis. All bands of the image must be array-valued and have the same dimensions.

Usage	Returns
`Image.arrayPad(lengths, `*`pad`*`)`	Image

Argument	Type	Details
this: `image`	Image	Array image to pad.
`lengths`	List	A list of desired lengths for each axis in the output arrays. Arrays are already as large or larger than the given length will be unchanged along that axis
`pad`	*Number, default: 0*	*The value to pad with.*

ee.Image.arrayProject

Projects the array in each pixel to a lower dimensional space by specifying the axes to retain. Dropped axes must be at most length 1.

Usage	Returns
`Image.arrayProject(axes)`	Image

Argument	Type	Details
this: `input`	Image	Input image.
`axes`	List	The axes to retain. Other axes will be discarded and must be at most length 1.

ee.Image.arrayReduce

Reduces elements of each array pixel.

Usage	Returns
`Image.arrayReduce(reducer, axes, `*`fieldAxis`*`)`	Image

Argument	Type	Details
this: `input`	Image	Input image.
`reducer`	Reducer	The reducer to apply
`axes`	List	The list of array axes to reduce in each pixel. The output will have a length of 1 in all these axes.
`fieldAxis`	*Integer, default: null*	*The axis for the reducer's input and output fields. Only required if the reducer has multiple inputs or outputs.*

ee.Image.arrayRepeat

Repeats each array pixel along the given axis. Each output pixel will have the shape of the input pixel, except length along the repeated axis, which will be multiplied by the number of copies.

Usage	Returns
`Image.arrayRepeat(axis, copies)`	Image

Argument	Type	Details
this: `input`	Image	Image of array pixels to be repeated.
`axis`	Integer	Axis along which to repeat each pixel's array.
`copies`	Image	Number of copies of each pixel.

ee.Image.arrayReshape

Converts array bands of an image with equally-shaped, possibly multidimensional pixels to an image of arrays with a new shape.

Usage	Returns
`Image.arrayReshape(lengths, dimensions)`	Image

Argument	Type	Details
this: `image`	Image	The image of arrays to reshape.
`lengths`	Image	A 1 band image specifying the new lengths of each axis of the input image specified as a 1-D array per pixel. There should be 'dimensions' lengths values in each shape' array. If one of the lengths is -1, then the corresponding length for that axis will be computed such that the total size remains constant. In particular, a shape of [-1] flattens into 1-D. At most one component of shape can be -1.
`dimensions`	Integer	The number of dimensions shared by all output array pixels.

ee.Image.arraySlice

Creates a subarray by slicing out each position along the given axis from the 'start' (inclusive) to 'end' (exclusive) by increments of 'step'. The result will have as many dimensions as the input, and the same length in all directions except the slicing axis, where the length will be the number of positions from 'start' to 'end' by 'step' that are in range of the input array's length along 'axis'. This means the result can be length 0 along the given axis if start=end, or if the start or end values are entirely out of range.

Usage	Returns
`Image.arraySlice(`*`axis`*`, `*`start`*`, `*`end`*`, `*`step`*`)`	Image

Argument	Type	Details
this: `input`	Image	Input array image.
`axis`	*Integer, default: 0*	*Axis to subset.*
`start`	*Image, default: null*	*The coordinate of the first slice (inclusive) along 'axis'. Negative numbers are used to position the start of slicing relative to the end of the array, where -1 starts at the last position on the axis, -2 starts at the next to last position, etc. There must one band for start indices, or one band per 'input' band. If this argument is not set or masked at some pixel, then the slice at that pixel will start at index 0.*
`end`	*Image, default: null*	*The coordinate (exclusive) at which to stop taking slices. By default this will be the length of the given axis. Negative numbers are used to position the end of slicing relative to the end of the array, where -1 will exclude the last position, -2 will exclude the last two positions, etc. There must be one band for end indices, or one band per 'input' band. If this argument is not set or masked at some pixel, then the slice at that pixel will end just after the last index.*
`step`	*Integer, default: 1*	*The separation between slices along 'axis'; a slice will be taken at each whole multiple of 'step' from 'start' (inclusive) to 'end' (exclusive). Must be positive.*

ee.Image.arraySort

Sorts elements of each array pixel along one axis.

Usage	Returns
`Image.arraySort(`*`keys`*`)`	Image

Argument	Type	Details
this: `image`	Image	Array image to sort.
`keys`	*Image, default: null*	*Optional keys to sort by. If not provided, the values are used as the keys. The keys can only have multiple elements along one axis, which determines the direction to sort in.*

ee.Image.arrayTranspose

Transposes two dimensions of each array pixel.

Usage	Returns
`Image.arrayTranspose(`*`axis1`*`, `*`axis2`*`)`	Image

Argument	Type	Details
this: `input`	Image	Input image.
`axis1`	*Integer, default: 0*	*First axis to swap.*
`axis2`	*Integer, default: 1*	*Second axis to swap.*

ee.Image.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`Image.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Image.asin

Computes the arc sine in radians of the input.

Usage	Returns
`Image.asin()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.atan

Computes the arc tangent in radians of the input.

Usage	Returns
`Image.atan()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.atan2

Calculates the angle formed by the 2D vector [x, y] for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is float.

Usage	Returns
`Image.atan2(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.bandNames

Returns a list containing the names of the bands of an image.

Usage	Returns
`Image.bandNames()`	List

Argument	Type	Details
this: `image`	Image	The image from which to get band names.

ee.Image.bandTypes

Returns a dictionary of the image's band types.

Usage	Returns
`Image.bandTypes()`	Dictionary

Argument	Type	Details
this: `image`	Image	The image from which to get band types.

ee.Image.bitCount

Calculates the number of one-bits in the 64-bit two's complement binary representation of the input.

Usage	Returns
`Image.bitCount()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.bitsToArrayImage

Turns the bits of an integer into a 1-D array. The array has a lengthup to the highest 'on' bit in the input.

Usage	Returns
`Image.bitsToArrayImage()`	Image

Argument	Type	Details
this: `input`	Image	Input image.

ee.Image.bitwiseAnd

Calculates the bitwise AND of the input values for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.bitwiseAnd(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.bitwiseNot

Calculates the bitwise NOT of the input, in the smallest signed integer type that can hold the input.

Usage	Returns
`Image.bitwiseNot()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.bitwiseOr

Calculates the bitwise OR of the input values for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.bitwiseOr(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.bitwiseXor

Calculates the bitwise XOR of the input values for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.bitwiseXor(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.blend

Overlays one image on top of another. The images are blended together using the masks as opacity. If either of images has only 1 band, it is replicated to match the number of bands in the other image.

Usage	Returns
`Image.blend(top)`	Image

Argument	Type	Details
this: `bottom`	Image	The bottom image.
`top`	Image	The top image.

ee.Image.byte

Casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Image.byte()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.cast

Casts some or all bands of an image to the specified types.

Usage	Returns
`Image.cast(bandTypes, bandOrder)`	Image

Argument	Type	Details
this: `image`	Image	The image to cast.
`bandTypes`	Dictionary	A dictionary from band name to band types. Types can be PixelTypes or strings. The valid strings are: 'int8', 'int16', 'int32', 'int64', 'uint8', 'uint16', 'uint32', 'byte', 'short', 'int', 'long', 'float' and 'double'. If bandTypes includes bands that are not already in the input image, they will be added to the image as transparent bands. If bandOrder isn't also specified, new bands will be appended in alphabetical order.
`bandOrder`	*List, default: null*	*A list specifying the order of the bands in the result. If specified, must match the full list of bands in the result.*

ee.Image.cat

Combines the given images into a single image which contains all bands from all of the images.

If two or more bands share a name, they are suffixed with an incrementing index.

The resulting image will have the metadata from the first input image, only.

This function will promote constant values into constant images.

Returns the combined image.

Usage	Returns
`ee.Image.cat(var_args)`	Image

Argument	Type	Details
`var_args`	VarArgs	The images to be combined.

ee.Image.cbrt

Computes the cubic root of the input.

Usage	Returns
`Image.cbrt()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.ceil

Computes the smallest integer greater than or equal to the input.

Usage	Returns
`Image.ceil()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.changeProj

Tweaks the projection of the input image, moving each pixel from its location in srcProj to the same coordinates in dstProj.

Usage	Returns
`Image.changeProj(srcProj, dstProj)`	Image

Argument	Type	Details
this: `input`	Image	
`srcProj`	Projection	The original projection.
`dstProj`	Projection	The new projection.

ee.Image.clamp

Clamps the values in all bands of an image to all lie within the specified range.

Usage	Returns
`Image.clamp(low, high)`	Image

Argument	Type	Details
this: `input`	Image	The image to clamp.
`low`	Float	The minimum allowed value in the range.
`high`	Float	The maximum allowed value in the range.

ee.Image.classify

Classifies an image.

Usage	Returns
`Image.classify(classifier, outputName)`	Image

Argument	Type	Details
this: `image`	Image	The image to classify. Bands are extracted from this image by name, and it must contain all the bands named in the classifier's schema.
`classifier`	Classifier	The classifier to use.
`outputName`	*String, default: "classification"*	*The name of the band to be added. If the classifier produces more than 1 output, this name is ignored.*

ee.Image.clip

Clips an image to a Geometry or Feature.

The output bands correspond exactly to the input bands, except data not covered by the geometry is masked. The output image retains the metadata of the input image.

Use clipToCollection to clip an image to a FeatureCollection.

Returns the clipped image.

Usage	Returns
`Image.clip(geometry)`	Image

Argument	Type	Details
this: `image`	Image	The Image instance.
`geometry`	Feature|Geometry|Object	The Geometry or Feature to clip to.

ee.Image.clipToBoundsAndScale

Clips an image to the bounds of a Geometry, and scales the clipped image to a particular size or scale.

Caution: providing a large or complex collection as the `geometry` argument can result in poor performance. Collating the geometry of collections does not scale well; use the smallest collection (or geometry) that is required to achieve the desired outcome.

Usage	Returns
`Image.clipToBoundsAndScale(`*`geometry`*`, `*`width`*`, `*`height`*`, `*`maxDimension`*`, `*`scale`*`)`	Image

Argument	Type	Details
this: `input`	Image	The image to clip and scale.
`geometry`	*Geometry, default: null*	*The Geometry to clip the image to. The image will be clipped to the bounding box, in the image's projection, of this geometry.*
`width`	*Integer, default: null*	*The width to scale the image to, in pixels. Must be provided along with "height". Exclusive with "maxDimension" and "scale".*
`height`	*Integer, default: null*	*The height to scale the image to, in pixels. Must be provided along with "width". Exclusive with "maxDimension" and "scale".*
`maxDimension`	*Integer, default: null*	*The maximum dimension to scale the image to, in pixels. Exclusive with "width", "height" and "scale".*
`scale`	*Float, default: null*	*If scale is specified, then the projection is scaled by dividing the specified scale value by the nominal size of a meter in the image's projection. Exclusive with "width", "height" and "maxDimension".*

ee.Image.clipToCollection

Clips an image to a FeatureCollection. The output bands correspond exactly the input bands, except data not covered by the geometry of at least one feature from the collection is masked. The output image retains the metadata of the input image.

Usage	Returns
`Image.clipToCollection(collection)`	Image

Argument	Type	Details
this: `input`	Image	The image to clip.
`collection`	Object	The FeatureCollection to clip to.

ee.Image.cluster

Applies a clusterer to an image. Returns a new image with a single band containing values from 0 to N, indicating the cluster each pixel is assigned to.

Usage	Returns
`Image.cluster(clusterer, outputName)`	Image

Argument	Type	Details
this: `image`	Image	The image to cluster. Must contain all the bands in the clusterer's schema.
`clusterer`	Clusterer	The clusterer to use.
`outputName`	*String, default: "cluster"*	*The name of the output band.*

ee.Image.connectedComponents

Finds connected components with the same value of the first band of the input and labels them with a globally unique value. Connectedness is specified by the given kernel. Objects larger than maxSize are considered background, and are masked.

Usage	Returns
`Image.connectedComponents(connectedness, maxSize)`	Image

Argument	Type	Details
this: `image`	Image	The image to label.
`connectedness`	Kernel	Connectedness kernel.
`maxSize`	Integer	Maximum size of objects to be labeled.

ee.Image.connectedPixelCount

Generate an image where each pixel contains the number of 4- or 8-connected neighbors (including itself).

Usage	Returns
`Image.connectedPixelCount(maxSize, eightConnected)`	Image

Argument	Type	Details
this: `input`	Image	The input image.

Argument	Type	Details
`maxSize`	*Integer, default: 100*	*The maximum size of the neighborhood in pixels.*
`eightConnected`	*Boolean, default: true*	*Whether to use 8-connected rather 4-connected rules.*

ee.Image.constant

Generates an image containing a constant value everywhere.

Usage	Returns
`ee.Image.constant(value)`	Image

Argument	Type	Details
`value`	Object	The value of the pixels in the constant image. Must be a number or an Array or a list of numbers or Arrays.

ee.Image.convolve

Convolves each band of an image with the given kernel.

Usage	Returns
`Image.convolve(kernel)`	Image

Argument	Type	Details
this: `image`	Image	The image to convolve.
`kernel`	Kernel	The kernel to convolve with.

ee.Image.copyProperties

Copies metadata properties from one element to another.

Usage	Returns
`Image.copyProperties(`*`source`*`, `*`properties`*`, `*`exclude`*`)`	Element

Argument	Type	Details
this: `destination`	*Element, default: null*	*The object whose properties to override.*
`source`	*Element, default: null*	*The object from which to copy the properties.*
`properties`	*List, default: null*	*The properties to copy. If omitted, all ordinary (i.e. non-system) properties are copied.*
`exclude`	*List, default: null*	*The list of properties to exclude when copying all properties. Must not be specified if properties is.*

ee.Image.cos

Computes the cosine of the input in radians.

Usage	Returns
`Image.cos()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.cosh

Computes the hyperbolic cosine of the input.

Usage	Returns
`Image.cosh()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.cumulativeCost

Computes a cumulative cost map based on an image containing costs to traverse each pixel and an image containing source locations.

Each output band represents the cumulative cost over the corresponding input cost band.

Usage	Returns
`Image.cumulativeCost(source, maxDistance, geodeticDistance)`	Image

Argument	Type	Details
this: `cost`	Image	An image representing the cost to traverse each pixel. Masked pixels can't be traversed. When comparing pixel traversal costs, we use band-wise dictionary ordering. Ancillary cost bands are only considered when paths over primary bands are equal cost.
`source`	Image	A single-band image representing the sources. A pixel value different from 0 defines a source pixel.
`maxDistance`	Float	Maximum distance for computation, in meters.
`geodeticDistance`	*Boolean, default: true*	*If true, geodetic distance along the curved surface is used, assuming a spherical Earth of radius 6378137.0. If false, euclidean distance in the 2D plane of the map projection is used (faster, but less accurate).*

ee.Image.date

Returns the acquisition time of an image as a Date object. This helper function is equivalent to `ee.Date(image.get('system:time_start'))`.

Usage	Returns
`Image.date()`	Date

Argument	Type	Details
this: `image`	Image	The image whose acquisition time to return.

ee.Image.derivative

Computes the X and Y discrete derivatives for each band in the input image, in pixel coordinates.

For each band of the input image, the output image will have two bands named with the suffixes `_x` and `_y`, containing the respective derivatives.

Usage	Returns
`Image.derivative()`	Image

Argument	Type	Details
this: `image`	Image	The input image.

ee.Image.digamma

Computes the digamma function of the input.

Usage	Returns
`Image.digamma()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.directionalDistanceTransform

For each zero-valued pixel in the source, get the distance to the nearest non-zero pixels in the given direction.

Returns a band of floating point distances called "distance".

Usage	Returns
`Image.directionalDistanceTransform(angle, maxDistance, `*`labelBand`*`)`	Image

Argument	Type	Details
this: `source`	Image	The source image.
`angle`	Float	The angle, in degrees, at which to search for non-zero pixels.
`maxDistance`	Integer	The maximum distance, in pixels, over which to search.
`labelBand`	*String, default: null*	*If provided, multi-band inputs are permitted and only this band is used for searching. All other bands are returned and populated with the per-band values found at the searched non-zero pixels in the label band.*

ee.Image.displace

Warps an image using an image of displacements.

Usage	Returns
`Image.displace(displacement, mode, maxOffset)`	Image

Argument	Type	Details
this: `image`	Image	The image to warp.
`displacement`	Image	An image containing displacement values. The first band is interpreted as the 'X' displacement and the second as the 'Y' displacement. Each displacement pixel is a [dx,dy] vector added to the pixel location to determine the corresponding pixel location in 'image'. Displacements are interpreted as meters in the default projection of the displacement image.
`mode`	*String, default: "bicubic"*	*The interpolation mode to use. One of 'nearest_neighbor', 'bilinear' or 'bicubic'.*
`maxOffset`	*Float, default: null*	*The maximum absolute offset in the displacement image. Providing this may improve processing performance.*

ee.Image.displacement

Determines displacements required to register an image to a reference image while allowing local, rubber sheet deformations. Displacements are computed in the CRS of the reference image, at a scale dictated by the lowest resolution of the following three projections: input image projection, reference image projection, and requested projection. The displacements are then transformed into the user-specified projection for output.

Usage	Returns
`Image.displacement(referenceImage, maxOffset, projection, patchWidth, stiffness)`	Image

Argument	Type	Details
this: `image`	Image	The image to register.
`referenceImage`	Image	The image to register to.
`maxOffset`	Float	The maximum offset allowed when attempting to align the input images, in meters. Using a smaller value can reduce computation time significantly, but it must still be large enough to cover the greatest displacement within the entire image region.

Argument	Type	Details
`projection`	*Projection, default: null*	*The projection in which to output displacement values. The default is the projection of the first band of the reference image.*
`patchWidth`	*Float, default: null*	*Patch size for detecting image offsets, in meters. This should be set large enough to capture texture, as well as large enough that ignorable objects are small within the patch. Default is null. Patch size will be determined automatically if not provided.*
`stiffness`	*Float, default: 5*	*Enforces a stiffness constraint on the solution. Valid values are in the range [0,10]. The stiffness is used for outlier rejection when determining displacements at adjacent grid points. Higher values move the solution towards a rigid transformation. Lower values allow more distortion or warping of the image during registration.*

ee.Image.distance

Computes the distance to the nearest non-zero pixel in each band, using the specified distance kernel.

Usage	Returns
`Image.distance(`*`kernel`*`, `*`skipMasked`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`kernel`	*Kernel, default: null*	*The distance kernel. One of chebyshev, euclidean, or manhattan.*
`skipMasked`	*Boolean, default: true*	*Mask output pixels if the corresponding input pixel is masked.*

ee.Image.divide

Divides the first value by the second, returning 0 for division by 0 for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.divide(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.double

Casts the input value to a 64-bit float.

Usage	Returns
`Image.double()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.entropy

Computes the windowed entropy for each band using the specified kernel centered on each input pixel. Entropy is computed as -sum(p * log2(p)), where p is the normalized probability of occurrence of the values encountered in each window.

Usage	Returns
`Image.entropy(kernel)`	Image

Argument	Type	Details
this: `image`	Image	The image for which to compute the entropy.
`kernel`	Kernel	A kernel specifying the window in which to compute.

ee.Image.eq

Returns 1 if and only if the first value is equal to the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order.

The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is boolean.

Usage	Returns
`Image.eq(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.erf

Computes the error function of the input.

Usage	Returns
`Image.erf()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.erfInv

Computes the inverse error function of the input.

Usage	Returns
`Image.erfInv()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.erfc

Computes the complementary error function of the input.

Usage	Returns
`Image.erfc()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.erfcInv

Computes the inverse complementary error function of the input.

Usage	Returns
`Image.erfcInv()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Image.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Image.exp

Computes the Euler's number e raised to the power of the input.

Usage	Returns
`Image.exp()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.expression

Evaluates an arithmetic expression on an image, possibly involving additional images.

The bands of the primary input image are available using the built-in function b(), as b(0) or b('band_name').

Variables in the expression are interpreted as additional image parameters which must be supplied in opt_map. The bands of each such image can be accessed like image.band_name or image[0].

Both b() and image[] allow multiple arguments, to specify multiple bands, such as b(1, 'name', 3). Calling b() with no arguments, or using a variable by itself, returns all bands of the image.

If the result of an expression is a single band, it can be assigned a name using the '=' operator (e.g.: x = a + b).

Returns the image computed by the provided expression.

Usage	Returns
`Image.expression(expression, `*`map`*`)`	Image

Argument	Type	Details
this: `image`	Image	The Image instance.
`expression`	String	The expression to evaluate.
`map`	*Dictionary, optional*	*A map of input images available by name.*

ee.Image.fastDistanceTransform

Returns the distance, as determined by the specified distance metric, to the nearest non-zero valued pixel in the input. The output contains values for all pixels within the given neighborhood size, regardless of the input's mask. Note: the default distance metric returns squared distance.

Usage	Returns
`Image.fastDistanceTransform(`*`neighborhood`*`, `*`units`*`, `*`metric`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`neighborhood`	*Integer, default: 256*	*Neighborhood size in pixels.*
`units`	*String, default: "pixels"*	*The units of the neighborhood, currently only 'pixels' are supported.*
`metric`	*String, default: "squared_euclidean"*	*Distance metric to use: options are `squared_euclidean`, `manhattan` or `chebyshev`.*

ee.Image.first

Selects the value of the first value for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.first(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.firstNonZero

Selects the first value if it is non-zero, and the second value otherwise for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in

the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.firstNonZero(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.float

Casts the input value to a 32-bit float.

Usage	Returns
`Image.float()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.floor

Computes the largest integer less than or equal to the input.

Usage	Returns
`Image.floor()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.focalMax

Applies a morphological reducer() filter to each band of an image using a named or custom kernel.

Usage	Returns
`Image.focalMax(`*`radius`*`, `*`kernelType`*`, `*`units`*`, `*`iterations`*`, `*`kernel`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to which to apply the operations.
`radius`	*Float, default: 1.5*	*The radius of the kernel to use.*
`kernelType`	*String, default: "circle"*	*The type of kernel to use. Options include: 'circle', 'square', 'cross', 'plus', octagon' and 'diamond'.*
`units`	*String, default: "pixels"*	*If a kernel is not specified, this determines whether the kernel is in meters or pixels.*
`iterations`	*Integer, default: 1*	*The number of times to apply the given kernel.*
`kernel`	*Kernel, default: null*	*A custom kernel. If used, kernelType and radius are ignored.*

ee.Image.focalMean

Applies a morphological mean filter to each band of an image using a named or custom kernel.

Usage	Returns
`Image.focalMean(`*`radius`*`, `*`kernelType`*`, `*`units`*`, `*`iterations`*`, `*`kernel`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to which to apply the operations.
`radius`	*Float, default: 1.5*	*The radius of the kernel to use.*
`kernelType`	*String, default: "circle"*	*The type of kernel to use. Options include: 'circle', 'square', 'cross', 'plus', octagon' and 'diamond'.*
`units`	*String, default: "pixels"*	*If a kernel is not specified, this determines whether the kernel is in meters or pixels.*
`iterations`	*Integer, default: 1*	*The number of times to apply the given kernel.*
`kernel`	*Kernel, default: null*	*A custom kernel. If used, kernelType and radius are ignored.*

ee.Image.focalMedian

Applies a morphological reducer() filter to each band of an image using a named or custom kernel.

Usage	Returns
`Image.focalMedian(`*`radius`*`, `*`kernelType`*`, `*`units`*`, `*`iterations`*`, `*`kernel`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to which to apply the operations.
`radius`	*Float, default: 1.5*	*The radius of the kernel to use.*
`kernelType`	*String, default: "circle"*	*The type of kernel to use. Options include: 'circle', 'square', 'cross', 'plus', octagon' and 'diamond'.*
`units`	*String, default: "pixels"*	*If a kernel is not specified, this determines whether the kernel is in meters or pixels.*
`iterations`	*Integer, default: 1*	*The number of times to apply the given kernel.*
`kernel`	*Kernel, default: null*	*A custom kernel. If used, kernelType and radius are ignored.*

ee.Image.focalMin

Applies a morphological reducer() filter to each band of an image using a named or custom kernel.

Usage	Returns
`Image.focalMin(`*`radius`*`, `*`kernelType`*`, `*`units`*`, `*`iterations`*`, `*`kernel`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to which to apply the operations.
`radius`	*Float, default: 1.5*	*The radius of the kernel to use.*
`kernelType`	*String, default: "circle"*	*The type of kernel to use. Options include: 'circle', 'square', 'cross', 'plus', octagon' and 'diamond'.*
`units`	*String, default: "pixels"*	*If a kernel is not specified, this determines whether the kernel is in meters or pixels.*
`iterations`	*Integer, default: 1*	*The number of times to apply the given kernel.*
`kernel`	*Kernel, default: null*	*A custom kernel. If used, kernelType and radius are ignored.*

ee.Image.focalMode

Applies a morphological reducer() filter to each band of an image using a named or custom kernel.

Usage	Returns
`Image.focalMode(`*`radius`*`, `*`kernelType`*`, `*`units`*`, `*`iterations`*`, `*`kernel`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to which to apply the operations.
`radius`	*Float, default: 1.5*	*The radius of the kernel to use.*
`kernelType`	*String, default: "circle"*	*The type of kernel to use. Options include: 'circle', 'square', 'cross', 'plus', octagon' and 'diamond'.*
`units`	*String, default: "pixels"*	*If a kernel is not specified, this determines whether the kernel is in meters or pixels.*
`iterations`	*Integer, default: 1*	*The number of times to apply the given kernel.*
`kernel`	*Kernel, default: null*	*A custom kernel. If used, kernelType and radius are ignored.*

ee.Image.gamma

Computes the gamma function of the input.

Usage	Returns
`Image.gamma()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.gammainc

Calculates the regularized lower incomplete Gamma function γ(x,a) for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is float.

Usage	Returns
`Image.gammainc(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.geometry

Returns the geometry of a given feature in a given projection.

Usage	Returns
`Image.geometry(`*`maxError`*`, `*`proj`*`, `*`geodesics`*`)`	Geometry

Argument	Type	Details
this: `feature`	Element	The feature from which the geometry is taken.
`maxError`	*ErrorMargin, default: null*	*The maximum amount of error tolerated when performing any necessary reprojection.*
`proj`	*Projection, default: null*	*If specified, the geometry will be in this projection. If unspecified, the geometry will be in its default projection.*
`geodesics`	*Boolean, default: null*	*If true, the geometry will have geodesic edges. If false, it will have edges as straight lines in the specified projection. If null, the edge interpretation will be the same as the original geometry. This argument is ignored if proj is not specified.*

ee.Image.get

Extract a property from a feature.

Usage	Returns
`Image.get(property)`	

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.Image.getArray

Extract a property from a feature.

Usage	Returns
`Image.getArray(property)`	Array

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.Image.getDownloadURL

Get a download URL for small chunks of image data in GeoTIFF or NumPy format. Maximum request size is 32 MB, maximum grid dimension is 10000.

Use getThumbURL for RGB visualization formats PNG and JPG.

Returns returns a download URL, or undefined if a callback was specified.

Usage	Returns
`Image.getDownloadURL(params, callback)`	Object|String

Argument	Type	Details
this: `image`	Image	The Image instance.
`params`	Object	An object containing download options with the following possible values: `name:` a base name to use when constructing filenames. Only applicable when format is "ZIPPED_GEO_TIFF" (default) or filePerBand is true. Defaults to the image id (or "download" for computed images) when format is "ZIPPED_GEO_TIFF" or filePerBand is true, otherwise a random character string is generated. Band names are appended when filePerBand is true.

Argument	Type	Details
		`bands:` a description of the bands to download. Must be an array of band names or an array of dictionaries, each with the following keys (optional parameters apply only when filePerBand is true): `id:` the name of the band, a string, required. `crs:` an optional CRS string defining the band projection. `crs_transform:` an optional array of 6 numbers specifying an affine transform from the specified CRS, in row-major order: [xScale, xShearing, xTranslation, yShearing, yScale, yTranslation] `dimensions:` an optional array of two integers defining the width and height to which the band is cropped. `scale:` an optional number, specifying the scale in meters of the band; ignored if crs and crs_transform are specified.
		`crs:` a default CRS string to use for any bands that do not explicitly specify one.
		`crs_transform:` a default affine transform to use for any bands that do not specify one, of the same format as the `crs_transform` of bands.
		`dimensions:` default image cropping dimensions to use for any bands that do not specify them.
		`scale:` a default scale to use for any bands that do not specify one; ignored if `crs` and `crs_transform` are specified.
		`region:` a polygon specifying a region to download; ignored if `crs` and `crs_transform` is specified.
		`filePerBand:` whether to produce a separate GeoTIFF per band (boolean). Defaults to true. If false, a single GeoTIFF is produced and all band-level transformations will be ignored.
		`format:` the download format. One of: "ZIPPED_GEO_TIFF" (GeoTIFF file(s) wrapped in a zip file, default) "GEO_TIFF" (GeoTIFF file) "NPY" (NumPy binary format) If "GEO_TIFF" or "NPY", filePerBand and all band-level transformations will be ignored. Loading a NumPy output results in a structured array.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Image.getInfo

An imperative function that returns information about this image via an AJAX call.

Returns a description of the image. Includes:

- bands - a list containing metadata about the bands in the collection.

- properties - a dictionary containing the image's metadata properties.

Usage	Returns
`Image.getInfo(`*`callback`*`)`	ImageDescription

Argument	Type	Details
this: `image`	Image	The Image instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. If supplied, will be called with the first parameter if successful and the second if unsuccessful.*

ee.Image.getMapId

An imperative function that returns a map ID and optional token, suitable for generating a Map overlay.

Returns an object which may be passed to ee.data.getTileUrl or ui.Map.addLayer. Undefined if a callback was specified.

Usage	Returns
`Image.getMapId(`*`visParams`*`, `*`callback`*`)`	MapId|Object

Argument	Type	Details
this: `image`	Image	The Image instance.
`visParams`	*ImageVisualizationParameters, optional*	*The visualization parameters.*
`callback`	*Function, optional*	*An async callback. If not supplied, the call is made synchronously.*

ee.Image.getNumber

Extract a property from a feature.

Usage	Returns
`Image.getNumber(property)`	Number

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.Image.getString

Extract a property from a feature.

Usage	Returns
`Image.getString(property)`	String

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.Image.getThumbId

Applies transformations and returns the thumbId.

Returns the thumb ID and optional token, or null if a callback is specified.

Usage	Returns
`Image.getThumbId(params, callback)`	ThumbnailId

Argument	Type	Details
this: `image`	Image	The Image instance.
`params`	Object	Parameters identical to ee.data.getMapId, plus, optionally: `dimensions` (a number or pair of numbers in format WIDTHxHEIGHT) Maximum dimensions of the thumbnail to render, in pixels. If only one number is passed, it is used as the maximum, and the other dimension is computed by proportional scaling. `region` Geospatial region of the image to render, it may be an ee.Geometry, GeoJSON, or an array of lat/lon points (E,S,W,N). If not set the default is the bounds image.

Argument	Type	Details
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Image.getThumbURL

Get a thumbnail URL for this image.

Returns a thumbnail URL, or undefined if a callback was specified.

Usage	Returns
`Image.getThumbURL(params, callback)`	Object|String

Argument	Type	Details
this: `image`	Image	The Image instance.
`params`	Object	Parameters identical to ee.data.getMapId, plus, optionally: `dimensions` (a number or pair of numbers in format WIDTHxHEIGHT) Maximum dimensions of the thumbnail to render, in pixels. If only one number is passed, it is used as the maximum, and the other dimension is computed by proportional scaling. `region` Geospatial region of the image to render, it may be an ee.Geometry, GeoJSON, or an array of lat/lon points (E,S,W,N). If not set the default is the bounds image. `format` (string) Either 'png' or 'jpg'.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Image.glcmTexture

Computes texture metrics from the Gray Level Co-occurrence Matrix around each pixel of every band. The GLCM is a tabulation of how often different combinations of pixel brightness values (grey levels) occur in an image. It counts the number of times a pixel of value X lies next to a pixel of value Y, in a particular direction and distance. and then derives statistics from this tabulation.

This implementation computes the 14 GLCM metrics proposed by Haralick, and 4 additional metrics from Conners. Inputs are required to be integer valued.

The output consists of 18 bands per input band if directional averaging is on and 18 bands per directional pair in the kernel, if not:

ASM: f1, Angular Second Moment; measures the number of repeated pairs

CONTRAST: f2, Contrast; measures the local contrast of an image

CORR: f3, Correlation; measures the correlation between pairs of pixels

VAR: f4, Variance; measures how spread out the distribution of gray-levels is

IDM: f5, Inverse Difference Moment; measures the homogeneity

SAVG: f6, Sum Average

SVAR: f7, Sum Variance

SENT: f8, Sum Entropy

ENT: f9, Entropy. Measures the randomness of a gray-level distribution

DVAR: f10, Difference variance

DENT: f11, Difference entropy

IMCORR1: f12, Information Measure of Corr. 1

IMCORR2: f13, Information Measure of Corr. 2

MAXCORR: f14, Max Corr. Coefficient. (not computed)

DISS: Dissimilarity

INERTIA: Inertia

SHADE: Cluster Shade

PROM: Cluster prominence

More information can be found in the two papers: Haralick et. al, 'Textural Features for Image Classification', http://doi.org/10.1109/TSMC.1973.4309314 and Conners, et al, Segmentation of a high-resolution urban scene using texture operators', http://doi.org/10.1016/0734-189X(84)90197-X.

Usage	Returns
`Image.glcmTexture(`*`size`*`, `*`kernel`*`, `*`average`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image for which to compute texture metrics.
`size`	*Integer, default: 1*	*The size of the neighborhood to include in each GLCM.*
`kernel`	*Kernel, default: null*	*A kernel specifying the x and y offsets over which to compute the GLCMs. A GLCM is computed for each pixel in the kernel that is non-zero, except the center pixel and as long as a GLCM hasn't already been computed for the same direction and distance. For example, if either or both of the east and west pixels are set, only 1 (horizontal) GLCM is computed. Kernels are scanned from left to right and top to bottom. The default is a 3x3 square, resulting in 4 GLCMs with the offsets (-1, -1), (0, -1), (1, -1) and (-1, 0).*
`average`	*Boolean, default: true*	*If true, the directional bands for each metric are averaged.*

ee.Image.gradient

Calculates the x and y gradient.

Usage	Returns
`Image.gradient()`	Image

Argument	Type	Details
this: `input`	Image	The input image.

ee.Image.gt

Returns 1 if and only if the first value is greater than the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is boolean.

Usage	Returns
`Image.gt(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.gte

Returns 1 if and only if the first value is greater than or equal to the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is boolean.

Usage	Returns
`Image.gte(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.hersDescriptor

Creates a dictionary of Histogram Error Ring Statistic (HERS) descriptor arrays from square array properties of an element. The HERS radius is taken to be the array's (side_length - 1) / 2.

Usage	Returns
`Image.hersDescriptor(`*`selectors`*`, `*`buckets`*`, `*`peakWidthScale`*`)`	Dictionary

Argument	Type	Details
this: `element`	Element	The element with array properties.
`selectors`	*List, default: null*	*The array properties for which descriptors will be created. Selected array properties must be square, floating point arrays. Defaults to all array properties.*
`buckets`	*Integer, default: 100*	*The number of HERS buckets. Defaults to 100.*

Argument	Type	Details
`peakWidthScale`	*Float, default: 1*	*The HERS peak width scale. Defaults to 1.0.*

ee.Image.hersFeature

Computes the Histogram Error Ring Statistic (HERS) for each pixel in each band matching the keys in the descriptor. Only the bands for which HERS could be computed are returned.

Usage	Returns
`Image.hersFeature(reference, `*`peakWidthScale`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`reference`	Dictionary	The reference descriptor computed with ee.Feature.hersDescriptor(...).
`peakWidthScale`	*Float, default: 1*	*The HERS peak width scale.*

ee.Image.hersImage

Computes the Histogram Error Ring Statistic (HERS) for each pair of pixels in each band present in both images. Only the bands for which HERS could be computed are returned.

Usage	Returns
`Image.hersImage(image2, radius, `*`buckets`*`, `*`peakWidthScale`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`image2`	Image	The image to compare.
`radius`	Integer	The radius of the window.
`buckets`	*Integer, default: 100*	*The number of HERS buckets.*
`peakWidthScale`	*Float, default: 1*	*The HERS peak width scale.*

ee.Image.hsvToRgb

Transforms the image from the HSV color space to the RGB color space. Expects a 3 band image in the range [0, 1], and produces three bands: red, green and blue with values in the range [0, 1].

Usage	Returns
`Image.hsvToRgb()`	Image

Argument	Type	Details
this: `image`	Image	The image to transform.

ee.Image.hypot

Calculates the magnitude of the 2D vector [x, y] for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is float.

Usage	Returns
`Image.hypot(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.id

Returns the ID of a given element within a collection. Objects outside collections are not guaranteed to have IDs.

Usage	Returns
`Image.id()`	String

Argument	Type	Details
this: `element`	Element	The element from which the ID is taken.

ee.Image.int

Casts the input value to a signed 32-bit integer.

Usage	Returns
`Image.int()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.int16

Casts the input value to a signed 16-bit integer.

Usage	Returns
`Image.int16()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.int32

Casts the input value to a signed 32-bit integer.

Usage	Returns
`Image.int32()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.int64

Casts the input value to a signed 64-bit integer.

Usage	Returns
`Image.int64()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.int8

Casts the input value to a signed 8-bit integer.

Usage	Returns
`Image.int8()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.interpolate

Interpolates each point in the first band of the input image into the piecewise-linear function specified by the x and y arrays. The x values must be strictly increasing. If an input point is less than the first or greater than the last x value, then the output is specified by the "behavior" argument: "extrapolate" specifies the output value is extrapolated from the two nearest points, "clamp" specifies the output value is taken from the nearest point, "input" specifies the output value is copied from the input and "mask" specifies the output value is masked.

Usage	Returns
`Image.interpolate(x, y, `*`behavior`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to which the interpolation is applied.
`x`	List	The x axis (input) values in the piecewise function.
`y`	List	The y axis (output) values in the piecewise function.
`behavior`	*String, default: "extrapolate"*	*The behavior for points that are outside of the range of the supplied function. Options are: 'extrapolate', 'clamp', 'mask' or 'input'.*

ee.Image.lanczos

Computes the Lanczos approximation of the input.

Usage	Returns
`Image.lanczos()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.leftShift

Calculates the left shift of v1 by v2 bits for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.leftShift(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.linkCollection

Links the source image to a matching image from an image collection.

Any specified bands or metadata will be added to the source image from the image found in the collection, and if the bands or metadata are already present they will be overwritten. If a matching image is not found, any new or updated bands will be fully masked and any new or updated metadata will be null. The output footprint will be the same as the source image footprint.

A match is determined if the source image and an image in the collection have a specific equivalent metadata property. If more than one collection image would match, the collection image selected is arbitrary. By default, images are matched on their 'system:index' metadata property.

This linking function is a convenience method for adding bands to a target image based on a specified shared metadata property and is intended to support linking collections that apply different processing/product generation to the same source imagery. For more expressive linking known as 'joining', see https://developers.google.com/earth-engine/guides/joins_intro.

Usage	Returns
`Image.linkCollection(imageCollection, linkedBands, linkedProperties, matchPropertyName)`	Image

Argument	Type	Details
this: `input`	Image	The source image a matching image in the collection will be linked to.
`imageCollection`	ImageCollection	The image collection searched to extract an image matching the source.
`linkedBands`	*Object, default: null*	*A band name or list of band names to add or update from the matching image.*
`linkedProperties`	*Object, default: null*	*A metadata property or list of properties to add or update from the matching image.*
`matchPropertyName`	*String, default: "system:index"*	*The metadata property name to use as a match criteria.*

ee.Image.load

Returns the image given its ID.

Usage	Returns
`ee.Image.load(id, version)`	Image

Argument	Type	Details
`id`	String	The asset ID of the image.
`version`	*Long, default: -1*	*The version of the asset. -1 signifies the latest version.*

ee.Image.loadGeoTIFF

Loads a GeoTIFF as an Image.

Usage	Returns
`ee.Image.loadGeoTIFF(uri)`	Image

Argument	Type	Details
`uri`	String	The Cloud Storage URI of the GeoTIFF to load.

ee.Image.log

Computes the natural logarithm of the input.

Usage	Returns
`Image.log()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.log10

Computes the base-10 logarithm of the input.

Usage	Returns
`Image.log10()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.long

Casts the input value to a signed 64-bit integer.

Usage	Returns
`Image.long()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.lt

Returns 1 if and only if the first value is less than the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is boolean.

Usage	Returns
`Image.lt(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.lte

Returns 1 if and only if the first value is less than or equal to the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in

the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is boolean.

Usage	Returns
`Image.lte(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.mask

Gets or sets an image's mask. The output image retains the metadata and footprint of the input image. Pixels where the mask changes from zero to another value will be filled with zeros, or the values closest to zero within the range of the pixel type.

Note: the version that sets a mask will be deprecated. To set a mask from an image on previously unmasked pixels, use Image.updateMask. To unmask previously masked pixels, use Image.unmask.

Usage	Returns
`Image.mask(`*`mask`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`mask`	*Image, default: null*	*The mask image. If specified, the input image is copied to the output but given the mask by the values of this image. If this is a single band, it is used for all bands in the input image. If not specified, returns an image created from the mask of the input image, scaled to the range [0:1] (invalid = 0, valid = 1.0).*

ee.Image.matrixCholeskyDecomposition

Calculates the Cholesky decomposition of a matrix. The Cholesky decomposition is a decomposition into the form L * L' where L is a lower triangular matrix. The input must be a symmetric positive-definite matrix. Returns an image with 1 band named 'L'.

Usage	Returns
`Image.matrixCholeskyDecomposition()`	Image

Argument	Type	Details
this: `image`	Image	Image of 2-D matrices to be decomposed.

ee.Image.matrixDeterminant

Computes the determinant of the matrix.

Usage	Returns
`Image.matrixDeterminant()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.matrixDiagonal

Computes the diagonal of the matrix in a single column.

Usage	Returns
`Image.matrixDiagonal()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.matrixFnorm

Computes the Frobenius norm of the matrix.

Usage	Returns
`Image.matrixFnorm()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.matrixIdentity

Creates an image where each pixel is a 2D identity matrix of the given size.

Usage	Returns
`ee.Image.matrixIdentity(size)`	Image

Argument	Type	Details
`size`	Integer	The length of each axis.

ee.Image.matrixInverse

Computes the inverse of the matrix.

Usage	Returns
`Image.matrixInverse()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.matrixLUDecomposition

Calculates the LU matrix decomposition such that P×input=L×U, where L is lower triangular (with unit diagonal terms), U is upper triangular and P is a partial pivot permutation matrix. The input matrix must be square. Returns an image with bands named 'L', 'U' and 'P'.

Usage	Returns
`Image.matrixLUDecomposition()`	Image

Argument	Type	Details
this: `image`	Image	Image of 2-D matrices to be decomposed.

ee.Image.matrixMultiply

Returns the matrix multiplication A * B for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.matrixMultiply(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.matrixPseudoInverse

Computes the Moore-Penrose pseudoinverse of the matrix.

Usage	Returns
`Image.matrixPseudoInverse()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.matrixQRDecomposition

Calculates the QR-decomposition of a matrix into two matrices Q and R such that input = QR, where Q is orthogonal, and R is upper triangular. Returns an image with bands named 'Q' and 'R'.

Usage	Returns
`Image.matrixQRDecomposition()`	Image

Argument	Type	Details
this: `image`	Image	Image of 2-D matrices to be decomposed.

ee.Image.matrixSingularValueDecomposition

Calculates the Singular Value Decomposition of the input matrix into U×S×V', such that U and V are orthogonal and S is diagonal. Returns an image with bands named 'U', 'S' and 'V'.

Usage	Returns
`Image.matrixSingularValueDecomposition()`	Image

Argument	Type	Details
this: `image`	Image	Image of 2-D matrices to be decomposed.

ee.Image.matrixSolve

Solves for x in the matrix equation A * x = B, finding a least-squares solution if A is overdetermined for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.matrixSolve(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.matrixToDiag

Computes a square diagonal matrix from a single column matrix.

Usage	Returns
`Image.matrixToDiag()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.matrixTrace

Computes the trace of the matrix.

Usage	Returns
`Image.matrixTrace()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.matrixTranspose

Transposes two dimensions of each array pixel.

Usage	Returns
`Image.matrixTranspose(`*`axis1`*`, `*`axis2`*`)`	Image

Argument	Type	Details
this: `input`	Image	Input image.
`axis1`	*Integer, default: 0*	*First axis to swap.*
`axis2`	*Integer, default: 1*	*Second axis to swap.*

ee.Image.max

Selects the maximum of the first and second values for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.max(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.medialAxis

Computes the discrete medial axis of the zero valued pixels of the first band of the input. Outputs 4 bands:

medial - the medial axis points, scaled by the distance.

coverage - the number of points supporting each medial axis point.

xlabel - the horizontal distance to the power point for each pixel.

ylabel - the vertical distance to the power point for each pixel.

Usage	Returns
`Image.medialAxis(`*`neighborhood`*`, `*`units`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`neighborhood`	*Integer, default: 256*	*Neighborhood size in pixels.*
`units`	*String, default: "pixels"*	*The units of the neighborhood, currently only 'pixels' are supported.*

ee.Image.metadata

Generates a constant image of type double from a metadata property.

Usage	Returns
`Image.metadata(property, name)`	Image

Argument	Type	Details
this: `image`	Image	The image from which to get the metadata
`property`	String	The property from which to take the value.
`name`	*String, default: null*	*The name for the output band. If unspecified, it will be the same as the property name.*

ee.Image.min

Selects the minimum of the first and second values for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.min(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.mod

Calculates the remainder of the first value divided by the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.mod(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.multiply

Multiplies the first value by the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.multiply(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.neighborhoodToArray

Turns the neighborhood of each pixel in a scalar image into a 2D array. Axes 0 and 1 of the output array correspond to Y and X axes of the image, respectively. The output image will have as many bands as the input; each output band has the same mask as the corresponding input band. The footprint and metadata of the input image are preserved.

Usage	Returns
`Image.neighborhoodToArray(kernel, `*`defaultValue`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to get pixels from; must be scalar-valued.
`kernel`	Kernel	The kernel specifying the shape of the neighborhood. Only fixed, square and rectangle kernels are supported. Weights are ignored; only the shape of the kernel is used.
`defaultValue`	*Float, default: 0*	*The value to use in the output arrays to replace the invalid (masked) pixels of the input. If the band type is integral, the fractional part of this value is discarded; in all cases, the value is clamped to the value range of the band.*

ee.Image.neighborhoodToBands

Turn the neighborhood of a pixel into a set of bands. The neighborhood is specified using a Kernel, and only non-zero-weight kernel values are used. The weights of the kernel is otherwise ignored.

Each input band produces x * y output bands. Each output band is named 'input_x_y' where x and y indicate the pixel's location in the kernel. For example, a 3x3 kernel operating on a 2-band image produces 18 output bands.

Usage	Returns
`Image.neighborhoodToBands(kernel)`	Image

Argument	Type	Details
this: `image`	Image	The image to get pixels from.
`kernel`	Kernel	The kernel specifying the neighborhood. Zero-weight values are ignored.

ee.Image.neq

Returns 1 if and only if the first value is not equal to the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is boolean.

Usage	Returns
`Image.neq(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.normalizedDifference

Computes the normalized difference between two bands. If the bands to use are not specified, uses the first two bands. The normalized difference is computed as (first − second) / (first + second). Note that the returned image band name is 'nd', the input image properties are not retained in the output image, and a negative pixel value in either input band will cause the output pixel to be masked. To avoid masking negative input values, use `ee.Image.expression()` to compute normalized difference.

Usage	Returns
`Image.normalizedDifference(`*`bandNames`*`)`	Image

Argument	Type	Details
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.*

ee.Image.not

Returns 0 if the input is non-zero, and 1 otherwise.

Usage	Returns
`Image.not()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.or

Returns 1 if and only if either input value is non-zero for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images

have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is boolean.

Usage	Returns
`Image.or(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.paint

Paints the geometries of a collection onto an image, using the given 'color' value to replace each band's values where any geometry covers the image (or, if a line width is specified, where the perimeters do).

This algorithm is most suitable for converting categorical data from feature properties to pixels in an image; if you wish to visualize a collection, consider using FeatureCollection.style instead, which supports RGB colors whereas this algorithm is strictly 'monochrome' (using single numeric values).

Usage	Returns
`Image.paint(featureCollection, color, width)`	Image

Argument	Type	Details
this: `image`	Image	The image on which the collection is painted.
`featureCollection`	FeatureCollection	The collection painted onto the image.
`color`	*Object, default: 0*	*The pixel value to paint into every band of the input image, either as a number which will be used for all features, or the name of a numeric property to take from each feature in the collection.*
`width`	*Object, default: null*	*Line width, either as a number which will be the line width for all geometries, or the name of a numeric property to take from each feature in the collection. If unspecified, the geometries will be filled instead of outlined.*

ee.Image.pixelArea

Generate an image in which the value of each pixel is the area of that pixel in square meters. The returned image has a single band called "area."

Usage	Returns
`ee.Image.pixelArea()`	Image

No arguments.

ee.Image.pixelCoordinates

Creates a two band image containing the x and y coordinates of each pixel in the given projection.

Usage	Returns
`ee.Image.pixelCoordinates(projection)`	Image

Argument	Type	Details
`projection`	Projection	The projection in which to provide pixels.

ee.Image.pixelLonLat

Creates an image with two bands named 'longitude' and 'latitude', containing the longitude and latitude at each pixel, in degrees.

Usage	Returns
`ee.Image.pixelLonLat()`	Image

No arguments.

ee.Image.polynomial

Compute a polynomial at each pixel using the given coefficients.

Usage	Returns
`Image.polynomial(coefficients)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`coefficients`	List	The polynomial coefficients in increasing order of degree starting with the constant term.

ee.Image.pow

Raises the first value to the power of the second for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is float.

Usage	Returns
`Image.pow(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.projection

Returns the default projection of an Image. Throws an error if the bands of the image don't all have the same projection.

Usage	Returns
`Image.projection()`	Projection

Argument	Type	Details
this: `image`	Image	The image from which to get the projection.

ee.Image.propertyNames

Returns the names of properties on this element.

Usage	Returns
`Image.propertyNames()`	List

Argument	Type	Details
this: `element`	Element	

ee.Image.random

Generates a random number at each pixel location. When using the 'uniform' distribution, outputs are in the range of [0 to 1). Using the 'normal' distribution, the outputs have μ=0, **σ**=1, but no explicit limits.

Usage	Returns
`ee.Image.random(`*`seed`*`, `*`distribution`*`)`	Image

Argument	Type	Details
`seed`	*Long, default: 0*	*Seed for the random number generator.*
`distribution`	*String, default: "uniform"*	*The distribution type of random numbers to produce. One of 'uniform' or 'normal'.*

ee.Image.randomVisualizer

Creates a visualization image by assigning a random color to each unique value of the pixels of the first band. The first three bands of the output image will contain 8-bit R, G and B values, followed by all bands of the input image.

Usage	Returns
`Image.randomVisualizer()`	Image

Argument	Type	Details
this: `image`	Image	Image with at least one band.

ee.Image.reduce

Applies a reducer to all of the bands of an image.

The reducer must have a single input and will be called at each pixel to reduce the stack of band values.

The output image will have one band for each reducer output.

Usage	Returns
`Image.reduce(reducer)`	Image

Argument	Type	Details
this: `image`	Image	The image to reduce.
`reducer`	Reducer	The reducer to apply to the given image.

ee.Image.reduceConnectedComponents

Applies a reducer to all of the pixels inside of each 'object'. Pixels are considered to belong to an object if they are connected (8-way) and have the same value in the 'label' band. The label band is only used to identify the connectedness; the rest are provided as inputs to the reducer.

Usage	Returns
`Image.reduceConnectedComponents(reducer, `*`labelBand`*`, `*`maxSize`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`reducer`	Reducer	The reducer to apply to pixels within the connected component.
`labelBand`	*String, default: null*	*The name of the band to use to detect connectedness. If unspecified, the first band is used.*
`maxSize`	*Integer, default: 256*	*Size of the neighborhood to consider when aggregating values. Any objects larger than maxSize in either the horizontal or vertical dimension will be masked, since portions of the object might be outside of the neighborhood.*

ee.Image.reduceNeighborhood

Applies the given reducer to the neighborhood around each pixel, as determined by the given kernel. If the reducer has a single input, it will be applied separately to each band of the collection; otherwise it must have the same number of inputs as the input image has bands.

The reducer output names determine the names of the output bands: reducers with multiple inputs will use the output names directly, while reducers with a single input will prefix the output name with the input band name (e.g. '10_mean', '20_mean', etc.).

Reducers with weighted inputs can have the input weight based on the input mask, the kernel value, or the smaller of those two.

Usage	Returns
`Image.reduceNeighborhood(reducer, kernel, inputWeight, skipMasked, optimization)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`reducer`	Reducer	The reducer to apply to pixels within the neighborhood.
`kernel`	Kernel	The kernel defining the neighborhood.
`inputWeight`	*String, default: "kernel"*	*One of 'mask', 'kernel', or 'min'.*
`skipMasked`	*Boolean, default: true*	*Mask output pixels if the corresponding input pixel is masked.*
`optimization`	*String, default: null*	*Optimization strategy. Options are 'boxcar' and 'window'. The 'boxcar' method is a fast method for computing count, sum or mean. It requires a homogeneous kernel, a single-input reducer and either MASK, KERNEL or no weighting. The 'window' method uses a running window, and has the same requirements as 'boxcar', but can use any single input reducer. Both methods require considerable additional memory.*

ee.Image.reduceRegion

Apply a reducer to all the pixels in a specific region.

Either the reducer must have the same number of inputs as the input image has bands, or it must have a single input and will be repeated for each band.

Returns a dictionary of the reducer's outputs.

Usage	Returns
`Image.reduceRegion(reducer, geometry, scale, crs, crsTransform, bestEffort, maxPixels, tileScale)`	Dictionary

Argument	Type	Details
this: `image`	Image	The image to reduce.
`reducer`	Reducer	The reducer to apply.
`geometry`	*Geometry, default: null*	*The region over which to reduce data. Defaults to the footprint of the image's first band.*
`scale`	*Float, default: null*	*A nominal scale in meters of the projection to work in.*
`crs`	*Projection, default: null*	*The projection to work in. If unspecified, the projection of the image's first band is used. If specified in addition to scale, rescaled to the specified scale.*
`crsTransform`	*List, default: null*	*The list of CRS transform values. This is a row-major ordering of the 3x2 transform matrix. This option is mutually exclusive with 'scale', and replaces any transform already set on the projection.*
`bestEffort`	*Boolean, default: false*	*If the polygon would contain too many pixels at the given scale, compute and use a larger scale which would allow the operation to succeed.*
`maxPixels`	*Long, default: 10000000*	*The maximum number of pixels to reduce.*
`tileScale`	*Float, default: 1*	*A scaling factor between 0.1 and 16 used to adjust aggregation tile size; setting a larger tileScale (e.g. 2 or 4) uses smaller tiles and may enable computations that run out of memory with the default.*

ee.Image.reduceRegions

Apply a reducer over the area of each feature in the given collection.

The reducer must have the same number of inputs as the input image has bands.

Returns the input features, each augmented with the corresponding reducer outputs.

Usage	Returns
`Image.reduceRegions(collection, reducer, scale, crs, crsTransform, tileScale)`	FeatureCollection

Argument	Type	Details
this: `image`	Image	The image to reduce.
`collection`	FeatureCollection	The features to reduce over.
`reducer`	Reducer	The reducer to apply.

Argument	Type	Details
`scale`	*Float, default: null*	*A nominal scale in meters of the projection to work in.*
`crs`	*Projection, default: null*	*The projection to work in. If unspecified, the projection of the image's first band is used. If specified in addition to scale, rescaled to the specified scale.*
`crsTransform`	*List, default: null*	*The list of CRS transform values. This is a row-major ordering of the 3x2 transform matrix. This option is mutually exclusive with 'scale', and will replace any transform already set on the projection.*
`tileScale`	*Float, default: 1*	*A scaling factor used to reduce aggregation tile size; using a larger tileScale (e.g. 2 or 4) may enable computations that run out of memory with the default.*

ee.Image.reduceResolution

Enables reprojection using the given reducer to combine all input pixels corresponding to each output pixel. If the reducer has a single input, it will be applied separately to each band of the collection; otherwise it must have the same number of inputs as the input image has bands.

The reducer output names determine the names of the output bands: reducers with multiple inputs will use the output names directly, reducers with a single input and single output will preserve the input band names, and reducers with a single input and multiple outputs will prefix the output name with the input band name (e.g. '10_mean', '10_stdDev', '20_mean', '20_stdDev', etc.).

Reducer input weights will be the product of the input mask and the fraction of the output pixel covered by the input pixel.

Usage	Returns
`Image.reduceResolution(reducer, bestEffort, maxPixels)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`reducer`	Reducer	The reducer to apply to be used for combining pixels.
`bestEffort`	*Boolean, default: false*	*If using the input at its default resolution would require too many pixels, start with already-reduced input pixels from a pyramid level that allows the operation to succeed.*
`maxPixels`	*Integer, default: 64*	*The maximum number of input pixels to combine for each output pixel. Setting this too large will cause out-of-memory problems.*

ee.Image.reduceToVectors

Convert an image to a feature collection by reducing homogeneous regions. Given an image containing a band of labeled segments and zero or more additional bands, runs a reducer over the pixels in each segment producing a feature per segment.

Either the reducer must have one fewer inputs than the image has bands, or it must have a single input and will be repeated for each band.

Usage	Returns
`Image.reduceToVectors(`*`reducer`*`, `*`geometry`*`, `*`scale`*`, `*`geometryType`*`, `*`eightConnected`*`, `*`labelProperty`*`, `*`crs`*`, `*`crsTransform`*`, `*`bestEffort`*`, `*`maxPixels`*`, `*`tileScale`*`, `*`geometryInNativeProjection`*`)`	FeatureCollection

Argument	Type	Details
this: `image`	Image	The input image. The first band is expected to be an integer type; adjacent pixels will be in the same segment if they have the same value in this band.
`reducer`	*Reducer, default: null*	*The reducer to apply. Its inputs will be taken from the image's bands after dropping the first band. Defaults to Reducer.countEvery()*
`geometry`	*Geometry, default: null*	*The region over which to reduce data. Defaults to the footprint of the image's first band.*
`scale`	*Float, default: null*	*A nominal scale in meters of the projection to work in.*
`geometryType`	*String, default: "polygon"*	*How to choose the geometry of each generated feature; one of 'polygon' (a polygon enclosing the pixels in the segment), 'bb' (a rectangle bounding the pixels), or 'centroid' (the centroid of the pixels).*
`eightConnected`	*Boolean, default: true*	*If true, diagonally-connected pixels are considered adjacent; otherwise only pixels that share an edge are.*
`labelProperty`	*String, default: "label"*	*If non-null, the value of the first band will be saved as the specified property of each feature.*
`crs`	*Projection, default: null*	*The projection to work in. If unspecified, the projection of the image's first band is used. If specified in addition to scale, rescaled to the specified scale.*
`crsTransform`	*List, default: null*	*The list of CRS transform values. This is a row-major ordering of the 3x2 transform matrix. This option is mutually exclusive with 'scale', and replaces any transform already set on the projection.*
`bestEffort`	*Boolean, default: false*	*If the polygon would contain too many pixels at the given scale, compute and use a larger scale which would allow the operation to succeed.*
`maxPixels`	*Long, default: 10000000*	*The maximum number of pixels to reduce.*
`tileScale`	*Float, default: 1*	*A scaling factor used to reduce aggregation tile size; using a larger tileScale (e.g. 2 or 4) may enable computations that run out of memory with the default.*

Argument	Type	Details
`geometryInNativeProjection`	*Boolean, default: false*	*Create geometries in the pixel projection, rather than WGS84.*

ee.Image.regexpRename

Renames the bands of an image by applying a regular expression replacement to the current band names. Any bands not matched by the regex will be copied over without renaming.

Usage	Returns
`Image.regexpRename(regex, replacement, all)`	Image

Argument	Type	Details
this: `input`	Image	The image containing the bands to rename.
`regex`	String	A regular expression to match in each band name.
`replacement`	String	The text with which to replace each match. Supports $n syntax for captured values.
`all`	*Boolean, default: true*	*If true, all matches in a given string will be replaced. Otherwise, only the first match in each string will be replaced.*

ee.Image.register

Registers an image to a reference image while allowing local, rubber sheet deformations. Displacements are computed in the CRS of the reference image, at a scale dictated by the lowest resolution of the following three projections: input image projection, reference image projection, and requested projection. The displacements then applied to the input image to register it with the reference.

Usage	Returns
`Image.register(referenceImage, maxOffset, patchWidth, stiffness)`	Image

Argument	Type	Details
this: `image`	Image	The image to register.
`referenceImage`	Image	The image to register to.

Argument	Type	Details
`maxOffset`	Float	The maximum offset allowed when attempting to align the input images, in meters. Using a smaller value can reduce computation time significantly, but it must still be large enough to cover the greatest displacement within the entire image region.
`patchWidth`	*Float, default: null*	*Patch size for detecting image offsets, in meters. This should be set large enough to capture texture, as well as large enough that ignorable objects are small within the patch. Default is null. Patch size will be determined automatically if notprovided.*
`stiffness`	*Float, default: 5*	*Enforces a stiffness constraint on the solution. Valid values are in the range [0,10]. The stiffness is used for outlier rejection when determining displacements at adjacent grid points. Higher values move the solution towards a rigid transformation. Lower values allow more distortion or warping of the image during registration.*

ee.Image.remap

Maps from input values to output values, represented by two parallel lists. Any input values not included in the input list are either set to defaultValue if it is given, or masked if it isn't. Note that inputs containing floating point values might sometimes fail to match due to floating point precision errors.

Usage	Returns
`Image.remap(from, to, `*`defaultValue`*`, `*`bandName`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to which the remapping is applied.
`from`	List	The source values (numbers or ee.Array). All values in this list will be mapped to the corresponding value in 'to'.
`to`	List	The destination values (numbers or ee.Array). These are used to replace the corresponding values in 'from'. Must have the same number of values as 'from'.
`defaultValue`	*Object, default: null*	*The default value to replace values that weren't matched by a value in 'from'. If not specified, unmatched values are masked out.*
`bandName`	*String, default: null*	*The name of the band to remap. If not specified, the first band in the image is used.*

ee.Image.rename

Rename the bands of an image.

Returns the renamed image.

Usage	Returns
`Image.rename(var_args)`	Image

Argument	Type	Details
this: `image`	Image	The Image instance.
`var_args`	List	The new names for the bands. Must match the number of bands in the Image.

ee.Image.reproject

Force an image to be computed in a given projection and resolution.

Usage	Returns
`Image.reproject(crs, crsTransform, scale)`	Image

Argument	Type	Details
this: `image`	Image	The image to reproject.
`crs`	Projection	The CRS to project the image to.
`crsTransform`	*List, default: null*	*The list of CRS transform values. This is a row-major ordering of the 3x2 transform matrix. This option is mutually exclusive with the scale option, and replaces any transform already on the projection.*
`scale`	*Float, default: null*	*If scale is specified, then the projection is scaled by dividing the specified scale value by the nominal size of a meter in the specified projection. If scale is not specified, then the scale of the given projection will be used.*

ee.Image.resample

An algorithm that returns an image identical to its argument, but which uses bilinear or bicubic interpolation (rather than the default nearest-neighbor) to compute pixels in projections other than its native projection or other levels of the same image pyramid.

This relies on the input image's default projection being meaningful, and so cannot be used on composites, for example. (Instead, you should resample the images that are used to create the composite.)

Usage	Returns
`Image.resample(`*`mode`*`)`	Image

Argument	Type	Details
this: `image`	Image	The Image to resample.
`mode`	*String, default: "bilinear"*	*The interpolation mode to use. One of 'bilinear' or 'bicubic'.)*

ee.Image.rgb

Create a 3-band image specifically for visualization. This uses the first band in each image.

Returns the combined image.

Usage	Returns
`ee.Image.rgb(r, g, b)`	Image

Argument	Type	Details
`r`	Image	The red image.
`g`	Image	The green image.
`b`	Image	The blue image.

ee.Image.rgbToHsv

Transforms the image from the RGB color space to the HSV color space. Expects a 3 band image in the range [0, 1], and produces three bands: hue, saturation and value with values in the range [0, 1].

Usage	Returns
`Image.rgbToHsv()`	Image

Argument	Type	Details
this: `image`	Image	The image to transform.

ee.Image.rightShift

Calculates the signed right shift of v1 by v2 bits for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.rightShift(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.round

Computes the integer nearest to the input.

Usage	Returns
`Image.round()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.rsedTransform

Computes the 2D maximal height surface created by placing an inverted parabola over each non-zero pixel of the input image, where the pixel's value is the height of the parabola. Viewed as a binary image (zero/not-zero) this is equivalent to buffering each non-zero input pixel by the square root of its value, in pixels.

Usage	Returns
`Image.rsedTransform(`*`neighborhood`*`, `*`units`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`neighborhood`	*Integer, default: 256*	*Neighborhood size in pixels.*
`units`	*String, default: "pixels"*	*The units of the neighborhood, currently only 'pixels' are supported.*

ee.Image.sample

Samples the pixels of an image, returning them as a FeatureCollection. Each feature will have 1 property per band in the input image. Note that the default behavior is to drop features that intersect masked pixels, which result in null-valued properties (see dropNulls argument).

Usage	Returns
`Image.sample(`*`region`*`, `*`scale`*`, `*`projection`*`, `*`factor`*`, `*`numPixels`*`, `*`seed`*`, `*`dropNulls`*`, `*`tileScale`*`, `*`geometries`*`)`	FeatureCollection

Argument	Type	Details
this: `image`	Image	The image to sample.
`region`	*Geometry, default: null*	*The region to sample from. If unspecified, uses the image's whole footprint.*
`scale`	*Float, default: null*	*A nominal scale in meters of the projection to sample in.*
`projection`	*Projection, default: null*	*The projection in which to sample. If unspecified, the projection of the image's first band is used. If specified in addition to scale, rescaled to the specified scale.*
`factor`	*Float, default: null*	*A subsampling factor, within (0, 1]. If specified, 'numPixels' must not be specified. Defaults to no subsampling.*
`numPixels`	*Long, default: null*	*The approximate number of pixels to sample. If specified, 'factor' must not be specified.*
`seed`	*Integer, default: 0*	*A randomization seed to use for subsampling.*
`dropNulls`	*Boolean, default: true*	*Post filter the result to drop features that have null-valued properties.*
`tileScale`	*Float, default: 1*	*A scaling factor used to reduce aggregation tile size; using a larger tileScale (e.g. 2 or 4) may enable computations that run out of memory with the default.*
`geometries`	*Boolean, default: false*	*If true, adds the center of the sampled pixel as the geometry property of the output feature. Otherwise, geometries will be omitted (saving memory).*

ee.Image.sampleRectangle

Extracts a rectangular region of pixels from an image into a ND array per band. The arrays are returned in a feature retaining the same properties as the image and a geometry the same as that used to sample the image (or the image footprint if unspecified). Each band is sampled in its input projection, and if no geometry is specified, sampled using its footprint. For scalar bands, the output array is 2D. For array bands the output array is (2+N)D where N is the number of dimensions in the original band. If sampling array bands, all arrays must have the same number of elements. If a band's sampled region is entirely masked and a default array value is specified, the default array value is used in-lieu of sampling the image.

Usage	Returns
`Image.sampleRectangle(`*`region`*`, `*`properties`*`, `*`defaultValue`*`, `*`defaultArrayValue`*`)`	Feature

Argument	Type	Details
this: `image`	Image	The image to sample.
`region`	*Geometry, default: null*	*The region whose projected bounding box is used to sample the image. Defaults to the footprint in each band.*
`properties`	*List, default: null*	*The properties to copy over from the sampled image. Defaults to all non-system properties.*
`defaultValue`	*Float, default: null*	*A default value used when a sampled pixel is masked or outside a band's footprint.*
`defaultArrayValue`	*Array, default: null*	*A default value used when a sampled array pixel is masked or outside a band's footprint.*

ee.Image.sampleRegions

Converts each pixel of an image (at a given scale) that intersects one or more regions to a Feature, returning them as a FeatureCollection. Each output feature will have one property per band of the input image, as well as any specified properties copied from the input feature.

Note that geometries will be snapped to pixel centers.

Usage	Returns
`Image.sampleRegions(collection, `*`properties`*`, `*`scale`*`, `*`projection`*`, `*`tileScale`*`, `*`geometries`*`)`	FeatureCollection

Argument	Type	Details
this: `image`	Image	The image to sample.

Argument	Type	Details
`collection`	FeatureCollection	The regions to sample over.
`properties`	*List, default: null*	*The list of properties to copy from each input feature. Defaults to all non-system properties.*
`scale`	*Float, default: null*	*A nominal scale in meters of the projection to sample in. If unspecified, the scale of the image's first band is used.*
`projection`	*Projection, default: null*	*The projection in which to sample. If unspecified, the projection of the image's first band is used. If specified in addition to scale, rescaled to the specified scale.*
`tileScale`	*Float, default: 1*	*A scaling factor used to reduce aggregation tile size; using a larger tileScale (e.g. 2 or 4) may enable computations that run out of memory with the default.*
`geometries`	*Boolean, default: false*	*If true, the results will include a point geometry per sampled pixel. Otherwise, geometries will be omitted (saving memory).*

ee.Image.select

Selects bands from an image.

Returns an image with the selected bands.

Usage	Returns
`Image.select(var_args)`	Image

Argument	Type	Details
this: `image`	Image	The Image instance.
`var_args`	VarArgs	One of two possibilities: • Any number of non-list arguments. All of these will be interpreted as band selectors. These can be band names, regexes, or numeric indices. E.g. selected = image.select('a', 'b', 3, 'd'); • Two lists. The first will be used as band selectors and the second as new names for the selected bands. The number of new names must match the number of selected bands. E.g. selected = image.select(['a', 4], ['newA', 'newB']);

ee.Image.selfMask

Updates an image's mask at all positions where the existing mask is not zero using the value of the image as the new mask value. The output image retains the metadata and footprint of the input image.

Usage	Returns
`Image.selfMask()`	Image

Argument	Type	Details
this: `image`	Image	The image to mask with itself.

ee.Image.serialize

Returns the serialized representation of this object.

Usage	Returns
`Image.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Image.set

Overrides one or more metadata properties of an Element.

Returns the element with the specified properties overridden.

Usage	Returns
`Image.set(var_args)`	Element

Argument	Type	Details
this: `element`	Element	The Element instance.
`var_args`	VarArgs	Either a dictionary of properties, or a vararg sequence of properties, e.g. key1, value1, key2, value2, ...

ee.Image.setDefaultProjection

Set a default projection to be applied to this image. The projection's resolution may be overridden by later operations.

Usage	Returns
`Image.setDefaultProjection(crs, crsTransform, scale)`	Image

Argument	Type	Details
this: `image`	Image	The image to reproject.
`crs`	Projection	The CRS to project the image to.
`crsTransform`	*List, default: null*	*The list of CRS transform values. This is a row-major ordering of the 3x2 transform matrix. This option is mutually exclusive with the scale option, and replaces any transform already on the projection.*
`scale`	*Float, default: null*	*If scale is specified, then the projection is scaled by dividing the specified scale value by the nominal size of a meter in the specified projection. If scale is not specified, then the scale of the given projection will be used.*

ee.Image.short

Casts the input value to a signed 16-bit integer.

Usage	Returns
`Image.short()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.signum

Computes the signum function (sign) of the input; zero if the input is zero, 1 if the input is greater than zero, -1 if the input is less than zero.

Usage	Returns
`Image.signum()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.sin

Computes the sine of the input in radians.

Usage	Returns
`Image.sin()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.sinh

Computes the hyperbolic sine of the input.

Usage	Returns
`Image.sinh()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.sldStyle

Styles a raster input with the provided OGC SLD styling.

Points of note:

* OGC SLD 1.0 and OGC SE 1.1 are supported.

* The XML document passed in can be complete, or just the SldRasterSymbolizer element and down.

* Exactly one SldRasterSymbolizer is required.

* Bands may be selected by their proper EarthEngine names or using numeric identifiers ("1", "2", ...). Proper EarthEngine names are tried first.

* The Histogram and Normalize contrast stretch mechanisms are supported.

* The type="values", type="intervals" and type="ramp" attributes for ColorMap element in SLD 1.0 (GeoServer extensions) are supported.

* Opacity is only taken into account when it is 0.0 (transparent). Non-zero opacity values are treated as completely opaque.

* The OverlapBehavior definition is currently ignored.

* The ShadedRelief mechanism is not currently supported.

* The ImageOutline mechanism is not currently supported.

* The Geometry element is ignored.

The output image will have histogram_bandname metadata if histogram equalization or normalization is requested.

Usage	Returns
`Image.sldStyle(sldXml)`	Image

Argument	Type	Details
this: `input`	Image	The image to rendering using the SLD.
`sldXml`	String	The OGC SLD 1.0 or 1.1 document (or fragment).

ee.Image.slice

Selects a contiguous group of bands from an image by position.

Usage	Returns
`Image.slice(start, `*`end`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image from which to select bands.
`start`	Integer	Where to start the selection. Negative numbers select from the end, counting backwards.

Argument	Type	Details
end	*Integer, default: null*	*Where to end the selection. If omitted, selects all bands from the start position to the end.*

ee.Image.spectralDilation

Computes the spectral/spatial dilation of an image by computing the spectral distance of each pixel under a structuring kernel from the centroid of all pixels under the kernel and taking the most distant result. See 'Spatial/spectral endmember extraction by multidimensional morphological operations.' IEEE transactions on geoscience and remote sensing 40.9 (2002): 2025-2041.

Usage	Returns
`Image.spectralDilation(`*`metric`*`, `*`kernel`*`, `*`useCentroid`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`metric`	*String, default: "sam"*	*The spectral distance metric to use. One of 'sam' (spectral angle mapper), 'sid' (spectral information divergence), 'sed' (squared euclidean distance), or 'emd' (earth movers distance).*
`kernel`	*Kernel, default: null*	*Connectedness kernel. Defaults to a square of radius 1 (8-way connected).*
`useCentroid`	*Boolean, default: false*	*If true, distances are computed from the mean of all pixels under the kernel instead of the kernel's center pixel.*

ee.Image.spectralDistance

Computes the per-pixel spectral distance between two images. If the images are array based then only the first band of each image is used; otherwise all bands are involved in the distance computation. The two images are therefore expected to contain the same number of bands or have the same 1-dimensional array length.

Usage	Returns
`Image.spectralDistance(image2, `*`metric`*`)`	Image

Argument	Type	Details
this: `image1`	Image	The first image.

Argument	Type	Details
`image2`	Image	The second image.
`metric`	*String, default: "sam"*	*The spectral distance metric to use. One of 'sam' (spectral angle mapper), 'sid' (spectral information divergence), 'sed' (squared euclidean distance), or 'emd' (earth movers distance).*

ee.Image.spectralErosion

Computes the spectral/spatial erosion of an image by computing the spectral distance of each pixel under a structuring kernel from the centroid of all pixels under the kernel and taking the closest result. See 'Spatial/spectral endmember extraction by multidimensional morphological operations.' IEEE transactions on geoscience and remote sensing 40.9 (2002): 2025-2041.

Usage	Returns
`Image.spectralErosion(`*`metric`*`, `*`kernel`*`, `*`useCentroid`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`metric`	*String, default: "sam"*	*The spectral distance metric to use. One of 'sam' (spectral angle mapper), 'sid' (spectral information divergence), 'sed' (squared euclidean distance), or 'emd' (earth movers distance).*
`kernel`	*Kernel, default: null*	*Connectedness kernel. Defaults to a square of radius 1 (8-way connected).*
`useCentroid`	*Boolean, default: false*	*If true, distances are computed from the mean of all pixels under the kernel instead of the kernel's center pixel.*

ee.Image.spectralGradient

Computes the spectral gradient over all bands of an image (or the first band if the image is Array typed) by computing the per-pixel difference between the spectral erosion and dilation with a given structuring kernel and distance metric. See: Plaza, Antonio, et al. 'Spatial/spectral endmember extraction by multidimensional morphological operations.' IEEE transactions on geoscience and remote sensing 40.9 (2002): 2025-2041.

Usage	Returns
`Image.spectralGradient(`*`metric`*`, `*`kernel`*`, `*`useCentroid`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`metric`	*String, default: "sam"*	*The spectral distance metric to use. One of 'sam' (spectral angle mapper), 'sid' (spectral information divergence), 'sed' (squared euclidean distance), or 'emd' (earth movers distance).*
`kernel`	*Kernel, default: null*	*Connectedness kernel. Defaults to a square of radius 1 (8-way connected).*
`useCentroid`	*Boolean, default: false*	*If true, distances are computed from the mean of all pixels under the kernel instead of the kernel's center pixel.*

ee.Image.sqrt

Computes the square root of the input.

Usage	Returns
`Image.sqrt()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.stratifiedSample

Extracts a stratified random sample of points from an image. Extracts the specified number of samples for each distinct value discovered within the 'classBand'. Returns a FeatureCollection of 1 Feature per extracted point, with each feature having 1 property per band in the input image. If there are less than the specified number of samples available for a given class value, then all of the points for that class will be included. Requires that the classBand contain integer values.

Usage	Returns
`Image.stratifiedSample(numPoints, `*`classBand`*`, `*`region`*`, `*`scale`*`, `*`projection`*`, `*`seed`*`, `*`classValues`*`, `*`classPoints`*`, `*`dropNulls`*`, `*`tileScale`*`, `*`geometries`*`)`	FeatureCollection

Argument	Type	Details
this: `image`	Image	The image to sample.

Argument	Type	Details
`numPoints`	Integer	The default number of points to sample in each class. Can be overridden for specific classes using the 'classValues' and 'classPoints' properties.
`classBand`	*String, default: null*	*The name of the band containing the classes to use for stratification. If unspecified, the first band of the input image is used.*
`region`	*Geometry, default: null*	*The region to sample from. If unspecified, the input image's whole footprint is used.*
`scale`	*Float, default: null*	*A nominal scale in meters of the projection to sample in. Defaults to the scale of the first band of the input image.*
`projection`	*Projection, default: null*	*The projection in which to sample. If unspecified, the projection of the input image's first band is used. If specified in addition to scale, rescaled to the specified scale.*
`seed`	*Integer, default: 0*	*A randomization seed to use for subsampling.*
`classValues`	*List, default: null*	*A list of class values for which to override the numPoints parameter. Must be the same size as classPoints or null.*
`classPoints`	*List, default: null*	*A list of the per-class maximum number of pixels to sample for each class in the classValues list. Must be the same size as classValues or null.*
`dropNulls`	*Boolean, default: true*	*Skip pixels in which any band is masked.*
`tileScale`	*Float, default: 1*	*A scaling factor used to reduce aggregation tile size; using a larger tileScale (e.g. 2 or 4) may enable computations that run out of memory with the default.*
`geometries`	*Boolean, default: false*	*If true, the results will include a geometry per sampled pixel. Otherwise, geometries will be omitted (saving memory).*

ee.Image.subtract

Subtracts the second value from the first for each matched pair of bands in image1 and image2. If either image1 or image2 has only 1 band, then it is used against all the bands in the other image. If the images have the same number of bands, but not the same names, they're used pairwise in the natural order. The output bands are named for the longer of the two inputs, or if they're equal in length, in image1's order. The type of the output pixels is the union of the input types.

Usage	Returns
`Image.subtract(image2)`	Image

Argument	Type	Details
this: `image1`	Image	The image from which the left operand bands are taken.
`image2`	Image	The image from which the right operand bands are taken.

ee.Image.tan

Computes the tangent of the input in radians.

Usage	Returns
`Image.tan()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.tanh

Computes the hyperbolic tangent of the input.

Usage	Returns
`Image.tanh()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toArray

Concatenates pixels from each band into a single array per pixel. The result will be masked if any input bands are masked.

Usage	Returns
`Image.toArray(`*`axis`*`)`	Image

Argument	Type	Details
this: `image`	Image	Image of bands to convert to an array per pixel. Bands must have scalar pixels, or array pixels with equal dimensionality.
`axis`	*Integer, default: 0*	*Axis to concatenate along; must be at least 0 and at most the dimension of the inputs. If the axis equals the dimension of the inputs, the result will have 1 more dimension than the inputs.*

ee.Image.toByte

Casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Image.toByte()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toDictionary

Extract properties from a feature as a dictionary.

Usage	Returns
`Image.toDictionary(`*`properties`*`)`	Dictionary

Argument	Type	Details
this: `element`	Element	The feature to extract the property from.
`properties`	*List, default: null*	*The list of properties to extract. Defaults to all non-system properties.*

ee.Image.toDouble

Casts the input value to a 64-bit float.

Usage	Returns
`Image.toDouble()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toFloat

Casts the input value to a 32-bit float.

Usage	Returns
`Image.toFloat()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toInt

Casts the input value to a signed 32-bit integer.

Usage	Returns
`Image.toInt()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toInt16

Casts the input value to a signed 16-bit integer.

Usage	Returns
`Image.toInt16()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toInt32

Casts the input value to a signed 32-bit integer.

Usage	Returns
`Image.toInt32()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toInt64

Casts the input value to a signed 64-bit integer.

Usage	Returns
`Image.toInt64()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toInt8

Casts the input value to a signed 8-bit integer.

Usage	Returns
`Image.toInt8()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toLong

Casts the input value to a signed 64-bit integer.

Usage	Returns
`Image.toLong()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toShort

Casts the input value to a signed 16-bit integer.

Usage	Returns
`Image.toShort()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toUint16

Casts the input value to an unsigned 16-bit integer.

Usage	Returns
`Image.toUint16()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toUint32

Casts the input value to an unsigned 32-bit integer.

Usage	Returns
`Image.toUint32()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.toUint8

Casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Image.toUint8()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.translate

Translate the input image.

Usage	Returns
`Image.translate(x, y, units, proj)`	Image

Argument	Type	Details
this: `input`	Image	
`x`	Float	
`y`	Float	
`units`	*String, default: "meters"*	*The units for x and y; 'meters' or 'pixels'.*
`proj`	*Projection, default: null*	*The projection in which to translate the image; defaults to the projection of the first band.*

ee.Image.trigamma

Computes the trigamma function of the input.

Usage	Returns
`Image.trigamma()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.uint16

Casts the input value to an unsigned 16-bit integer.

Usage	Returns
`Image.uint16()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.uint32

Casts the input value to an unsigned 32-bit integer.

Usage	Returns
`Image.uint32()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.uint8

Casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Image.uint8()`	Image

Argument	Type	Details
this: `value`	Image	The image to which the operation is applied.

ee.Image.unitScale

Scales the input so that the range of input values [low, high] becomes [0, 1]. Values outside the range are NOT clamped. This algorithm always produces floating point pixels.

Usage	Returns
`Image.unitScale(low, high)`	Image

Argument	Type	Details
this: `input`	Image	The image to scale.
`low`	Float	The value mapped to 0.
`high`	Float	The value mapped to 1.

ee.Image.unmask

Replaces mask and value of the input image with the mask and value of another image at all positions where the input mask is zero. The output image retains the metadata of the input image. By default, the output image also retains the footprint of the input, but setting sameFootprint to false allows to extend the footprint.

Usage	Returns
`Image.unmask(`*`value`*`, `*`sameFootprint`*`)`	Image

Argument	Type	Details
this: `input`	Image	Input image.
`value`	*Image, default: null*	*New value and mask for the masked pixels of the input image. If not specified, defaults to constant zero image which is valid everywhere.*
`sameFootprint`	*Boolean, default: true*	*If true (or unspecified), the output retains the footprint of the input image. If false, the footprint of the output is the union of the input footprint with the footprint of the value image.*

ee.Image.unmix

Unmix each pixel with the given endmembers, by computing the pseudo-inverse and multiplying it through each pixel. Returns an image of doubles with the same number of bands as endmembers.

Usage	Returns
`Image.unmix(endmembers, `*`sumToOne`*`, `*`nonNegative`*`)`	Image

Argument	Type	Details
this: `image`	Image	The input image.
`endmembers`	List	The endmembers to unmix with.
`sumToOne`	*Boolean, default: false*	*Constrain the outputs to sum to one.*
`nonNegative`	*Boolean, default: false*	*Constrain the outputs to be non-negative.*

ee.Image.updateMask

Updates an image's mask at all positions where the existing mask is not zero. The output image retains the metadata and footprint of the input image.

Usage	Returns
`Image.updateMask(mask)`	Image

Argument	Type	Details
this: `image`	Image	Input image.
`mask`	Image	New mask for the image, as a floating-point value in the range [0, 1] (invalid = 0, valid = 1). If this image has a single band, it is used for all bands in the input image; otherwise, must have the same number of bands as the input image.

ee.Image.visualize

Produces an RGB or grayscale visualization of an image. Each of the gain, bias, min, max and gamma arguments can take either a single value, which will be applied to all bands, or a list of values the same length as bands.

Usage	Returns
`Image.visualize(`*`bands`*`, `*`gain`*`, `*`bias`*`, `*`min`*`, `*`max`*`, `*`gamma`*`, `*`opacity`*`, `*`palette`*`, `*`forceRgbOutput`*`)`	Image

Argument	Type	Details
this: `image`	Image	The image to visualize.
`bands`	*Object, default: null*	*A list of the bands to visualize. If empty, the first 3 are used.*
`gain`	*Object, default: null*	*The visualization gain(s) to use.*
`bias`	*Object, default: null*	*The visualization bias(es) to use.*
`min`	*Object, default: null*	*The value(s) to map to RGB8 value 0.*
`max`	*Object, default: null*	*The value(s) to map to RGB8 value 255.*
`gamma`	*Object, default: null*	*The gamma correction factor(s) to use.*
`opacity`	*Number, default: null*	*The opacity scaling factor to use.*

Argument	Type	Details
`palette`	*Object, default: null*	*The color palette to use. List of CSS color identifiers or hexadecimal color strings (e.g. ['red', '00FF00', 'bluevlolet']).*
`forceRgbOutput`	*Boolean, default: false*	*Whether to produce RGB output even for single-band inputs.*

ee.Image.where

Performs conditional replacement of values.

For each pixel in each band of 'input', if the corresponding pixel in 'test' is nonzero, output the corresponding pixel in value, otherwise output the input pixel.

If at a given pixel, either test or value is masked, the input value is used. If the input is masked, nothing is done.

The output bands have the same names as the input bands. The output type of each band is the larger of the input and value types. The output image retains the metadata and footprint of the input image.

Usage	Returns
`Image.where(test, value)`	Image

Argument	Type	Details
this: `input`	Image	The input image.
`test`	Image	The test image. The pixels of this image determines which of the input pixels is returned. If this is a single band, it is used for all bands in the input image. This may not be an array image.
`value`	Image	The output value to use where test is not zero. If this is a single band, it is used for all bands in the input image.

ee.Image.zeroCrossing

Finds zero-crossings on each band of an image.

Usage	Returns
`Image.zeroCrossing()`	Image

Argument	Type	Details
this: `image`	Image	The image from which to compute zero crossings.

ee.ImageCollection

ImageCollections can be constructed from the following arguments:

- A string: assumed to be the name of a collection,

- A list of images, or anything that can be used to construct an image.

- A single image.

- A computed object - reinterpreted as a collection.

Usage	Returns
`ee.ImageCollection(args)`	ImageCollection

Argument	Type	Details
`args`	ComputedObject|Image|List	The constructor arguments.

ee.ImageCollection.aggregate_array

Aggregates over a given property of the objects in a collection, calculating a list of all the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_array(property)`	List

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_count

Aggregates over a given property of the objects in a collection, calculating the number of non-null values of the property.

Usage	Returns
`ImageCollection.aggregate_count(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_count_distinct

Aggregates over a given property of the objects in a collection, calculating the number of distinct values for the selected property.

Usage	Returns
`ImageCollection.aggregate_count_distinct(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_first

Aggregates over a given property of the objects in a collection, calculating the property value of the first object in the collection.

Usage	Returns
`ImageCollection.aggregate_first(property)`	

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_histogram

Aggregates over a given property of the objects in a collection, calculating a histogram of the selected property.

Usage	Returns
`ImageCollection.aggregate_histogram(property)`	Dictionary

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_max

Aggregates over a given property of the objects in a collection, calculating the maximum of the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_max(property)`	

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_mean

Aggregates over a given property of the objects in a collection, calculating the mean of the selected property.

Usage	Returns
`ImageCollection.aggregate_mean(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.

Argument	Type	Details
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_min

Aggregates over a given property of the objects in a collection, calculating the minimum of the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_min(property)`	

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_product

Aggregates over a given property of the objects in a collection, calculating the product of the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_product(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_sample_sd

Aggregates over a given property of the objects in a collection, calculating the sample std. deviation of the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_sample_sd(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_sample_var

Aggregates over a given property of the objects in a collection, calculating the sample variance of the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_sample_var(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_stats

Aggregates over a given property of the objects in a collection, calculating the sum, min, max, mean, sample standard deviation, sample variance, total standard deviation and total variance of the selected property.

Usage	Returns
`ImageCollection.aggregate_stats(property)`	Dictionary

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_sum

Aggregates over a given property of the objects in a collection, calculating the sum of the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_sum(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_total_sd

Aggregates over a given property of the objects in a collection, calculating the total std. deviation of the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_total_sd(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.aggregate_total_var

Aggregates over a given property of the objects in a collection, calculating the total variance of the values of the selected property.

Usage	Returns
`ImageCollection.aggregate_total_var(property)`	Number

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`property`	String	The property to use from each element of the collection.

ee.ImageCollection.and

Reduces an image collection by setting each pixel to 1 iff all the non-masked values at that pixel are non-zero across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.and()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`ImageCollection.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.

Argument	Type	Details
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.ImageCollection.cast

Casts some or all bands of each image in an ImageCollection to the specified types.

Usage	Returns
`ImageCollection.cast(bandTypes, bandOrder)`	ImageCollection

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to cast.
`bandTypes`	Dictionary	A dictionary from band name to band types. Types can be PixelTypes or strings. The valid strings are: 'int8', 'int16', 'int32', 'int64', 'uint8', 'uint16', 'uint32', 'byte', 'short', 'int', 'long', 'float' and 'double'. Must include all bands already in any image in the collection. If this includes bands that are not already in an input image, they will be added to the image as transparent bands.
`bandOrder`	List	A list specifying the order of the bands in the result.Must match the keys of bandTypes.

ee.ImageCollection.combine

Makes a new collection that is a copy of the images in primary, adding all the bands from the image in secondary with a matching ID. If there are no matching IDs, the resulting collection will be empty. This is equivalent to an inner join on ID with merging of the bands of the result.

Note that this algorithm assumes that for a matching pair of inputs, both have the same footprint and metadata.

Usage	Returns
`ImageCollection.combine(secondary, `*`overwrite`*`)`	ImageCollection

Argument	Type	Details
this: `primary`	ImageCollection	The primary collection to join.

Argument	Type	Details
`secondary`	ImageCollection	The secondary collection to join.
`overwrite`	*Boolean, default: false*	*If true, bands with the same name will get overwritten. If false, bands with the same name will be renamed.*

ee.ImageCollection.copyProperties

Copies metadata properties from one element to another.

Usage	Returns
`ImageCollection.copyProperties(`*`source`*`, `*`properties`*`, `*`exclude`*`)`	Element

Argument	Type	Details
this: `destination`	*Element, default: null*	*The object whose properties to override.*
`source`	*Element, default: null*	*The object from which to copy the properties.*
`properties`	*List, default: null*	*The properties to copy. If omitted, all ordinary (i.e. non-system) properties are copied.*
`exclude`	*List, default: null*	*The list of properties to exclude when copying all properties. Must not be specified if properties is.*

ee.ImageCollection.count

Reduces an image collection by calculating the number of images with a valid mask at each pixel across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.count()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.distance

Produces a DOUBLE image where each pixel is the distance in meters from the pixel center to the nearest Point, LineString, or polygonal boundary in the collection. Note distance is also measured within interiors of polygons. Pixels that are not within 'searchRadius' meters of a geometry will be masked out.

Distances are computed on a sphere, so there is a small error proportional to the latitude difference between each pixel and the nearest geometry.

Usage	Returns
`ImageCollection.distance(`*`searchRadius`*`, `*`maxError`*`)`	Image

Argument	Type	Details
this: `features`	FeatureCollection	Feature collection from which to get features used to compute pixel distances.
`searchRadius`	*Float, default: 100000*	*Maximum distance in meters from each pixel to look for edges. Pixels will be masked unless there are edges within this distance.*
`maxError`	*Float, default: 100*	*Maximum reprojection error in meters, only used if the input polylines require reprojection. If '0' is provided, then this operation will fail if projection is required.*

ee.ImageCollection.distinct

Removes duplicates from a collection. Note that duplicates are determined using a strong hash over the serialized form of the selected properties.

Usage	Returns
`ImageCollection.distinct(properties)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection from which objects will be selected.
`properties`	Object	A property name or a list of property names to use for comparison. The '.geo' property can be included to compare object geometries.

ee.ImageCollection.draw

Paints a vector collection for visualization. Not intended for use as input to other algorithms.

Usage	Returns
`ImageCollection.draw(color, pointRadius, strokeWidth)`	Image

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to draw.
`color`	String	A hex string in the format RRGGBB specifying the color to use for drawing the features.
`pointRadius`	*Integer, default: 3*	*The radius in pixels of the point markers.*
`strokeWidth`	*Integer, default: 2*	*The width in pixels of lines and polygon borders.*

ee.ImageCollection.errorMatrix

Computes a 2D error matrix for a collection by comparing two columns of a collection: one containing the actual values, and one containing predicted values.The values are expected to be small contiguous integers, starting from 0. Axis 0 (the rows) of the matrix correspond to the actual values, and Axis 1 (the columns) to the predicted values.

Usage	Returns
`ImageCollection.errorMatrix(actual, predicted, order)`	ConfusionMatrix

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection.
`actual`	String	The name of the property containing the actual value.
`predicted`	String	The name of the property containing the predicted value.
`order`	*List, default: null*	*A list of the expected values. If this argument is not specified, the values are assumed to be contiguous and span the range 0 to maxValue. If specified, only values matching this list are used, and the matrix will have dimensions and order matching the this list.*

ee.ImageCollection.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`ImageCollection.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.ImageCollection.filter

Apply a filter to this collection.

Returns the filtered collection.

Usage	Returns
`ImageCollection.filter(filter)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`filter`	Filter	A filter to apply to this collection.

ee.ImageCollection.filterBounds

Shortcut to filter a collection by intersection with geometry. Items in the collection with a footprint that fails to intersect the given geometry will be excluded.

This is equivalent to this.filter(ee.Filter.bounds(...)).

Caution: providing a large or complex collection as the **`geometry`** argument can result in poor performance. Collating the geometry of collections does not scale well; use the smallest collection (or geometry) that is required to achieve the desired outcome.

Returns the filtered collection.

Usage	Returns
`ImageCollection.filterBounds(geometry)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`geometry`	ComputedObject|FeatureCollection|Geometry	The geometry, feature or collection to intersect with.

ee.ImageCollection.filterDate

Shortcut to filter a collection by a date range. The start and end may be Dates, numbers (interpreted as milliseconds since 1970-01-01T00:00:00Z), or strings (such as '1996-01-01T08:00'). Based on 'system:time_start'.

This is equivalent to this.filter(ee.Filter.date(...)); see the ee.Filter type for other date filtering options.

Returns the filtered collection.

Usage	Returns
`ImageCollection.filterDate(start, `*`end`*`)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`start`	Date|Number|String	The start date (inclusive).
end	*Date|Number|String, optional*	*The end date (exclusive). Optional. If not specified, a 1-millisecond range starting at 'start' is created.*

ee.ImageCollection.first

Returns the first entry from a given collection.

Usage	Returns
`ImageCollection.first()`	Image

Argument	Type	Details
this: `imagecollection`	ImageCollection	The ImageCollection instance.

ee.ImageCollection.flatten

Flattens collections of collections.

Usage	Returns
`ImageCollection.flatten()`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection of collections.

ee.ImageCollection.formaTrend

Computes the long and short term trends of a time series or optionally, the trends of the ratio of the time series and a covariate. The long term trend is estimated from the linear term of a regression on the full time series. The short term trend is computed as the windowed minimum over the time series.

The time series and covariate series are expected to contain a single band each, and the time series is expected to be evenly spaced in time. The output is 4 float bands: the long and short term trends, the t-test of the long term trend against the time series, and the Bruce Hansen test of parameter stability.

Usage	Returns
`ImageCollection.formaTrend(`*`covariates`*`, `*`windowSize`*`)`	Image

Argument	Type	Details
this: `timeSeries`	ImageCollection	Collection from which to extract trends.
`covariates`	*ImageCollection, default: null*	*Cofactors to use in the trend analysis.*
`windowSize`	*Integer, default: 6*	*Short term trend analysis window size, in images.*

ee.ImageCollection.fromImages

Returns the image collection containing the given images.

Usage	Returns
`ee.ImageCollection.fromImages(images)`	ImageCollection

Argument	Type	Details
`images`	List	The images to include in the collection.

ee.ImageCollection.geometry

Extracts and merges the geometries of a collection. Requires that all the geometries in the collection share the projection and edge interpretation.

Caution: providing a large or complex collection as input can result in poor performance. Collating the geometry of collections does not scale well; use the smallest collection that is required to achieve the desired outcome.

Usage	Returns
`ImageCollection.geometry(`*`maxError`*`)`	Geometry

Argument	Type	Details
this: `collection`	FeatureCollection	The collection whose geometries will be extracted.
`maxError`	*ErrorMargin, optional*	*An error margin to use when merging geometries.*

ee.ImageCollection.get

Extract a property from a feature.

Usage	Returns
`ImageCollection.get(property)`	

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.

Argument	Type	Details
`property`	String	The property to extract.

ee.ImageCollection.getArray

Extract a property from a feature.

Usage	Returns
`ImageCollection.getArray(property)`	Array

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.ImageCollection.getFilmstripThumbURL

Get the URL of a tiled thumbnail for this ImageCollection.

Returns a thumbnail URL, or undefined if a callback was specified.

Usage	Returns
`ImageCollection.getFilmstripThumbURL(params, callback)`	Object|String

Argument	Type	Details
this: `imagecollection`	ImageCollection	The ImageCollection instance.
`params`	Object	Parameters identical to ee.data.getMapId, plus, optionally: `dimensions` (a number or pair of numbers in format WIDTHxHEIGHT) Maximum dimensions of each thumbnail frame to render, in pixels. If only one number is passed, it is used as the maximum, and the other dimension is computed by proportional scaling. `region` (E,S,W,N or GeoJSON) Geospatial region of the image to render. By default, the whole image. `format` (string) Encoding format. Only 'png' or 'jpg' are accepted.

Argument	Type	Details
`callback`	*Function, optional*	*An optional callback which handles the resulting URL string. If not supplied, the call is made synchronously.*

ee.ImageCollection.getInfo

An imperative function that returns all the known information about this collection via an AJAX call.

Returns a collection description whose fields include:

- features: a list containing metadata about the images in the collection.

- bands: a dictionary describing the bands of the images in this collection.

- properties: an optional dictionary containing the collection's metadata properties.

Usage	Returns
`ImageCollection.getInfo(`*`callback`*`)`	ImageCollectionDescription

Argument	Type	Details
this: `imagecollection`	ImageCollection	The ImageCollection instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. If supplied, will be called with the first parameter if successful and the second if unsuccessful.*

ee.ImageCollection.getMapId

An imperative function that returns a map ID via a synchronous AJAX call.

This mosaics the collection to a single image and return a map ID suitable for building a Google Maps overlay.

Returns returns a map ID and optional token, which may be passed to ee.data.getTileUrl or ui.Map.addLayer. Undefined if a callback was specified.

Usage	Returns
`ImageCollection.getMapId(`*`visParams`*`, `*`callback`*`)`	MapId|Object

Argument	Type	Details
this: `imagecollection`	ImageCollection	The ImageCollection instance.
`visParams`	*Object, optional*	*The visualization parameters.*
`callback`	*Function, optional*	*An async callback. If not supplied, the call is made synchronously.*

ee.ImageCollection.getNumber

Extract a property from a feature.

Usage	Returns
`ImageCollection.getNumber(property)`	Number

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.ImageCollection.getRegion

Output an array of values for each [pixel, band, image] tuple in an ImageCollection. The output contains rows of id, lon, lat, time, and all bands for each image that intersects each pixel in the given region. Attempting to extract more than 1048576 values will result in an error.

Usage	Returns
`ImageCollection.getRegion(geometry, scale, crs, crsTransform)`	List

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to extract data from.
`geometry`	Geometry	The region over which to extract data.
`scale`	*Float, default: null*	*A nominal scale in meters of the projection to work in.*
`crs`	*Projection, optional*	*The projection to work in. If unspecified, defaults to EPSG:4326. If specified in addition to scale, the projection is rescaled to the specified scale.*

Argument	Type	Details
`crsTransform`	*List, default: null*	*The array of CRS transform values. This is a row-major ordering of a 3x2 affine transform. This option is mutually exclusive with the scale option, and will replace any transform already set on the given projection.*

ee.ImageCollection.getString

Extract a property from a feature.

Usage	Returns
`ImageCollection.getString(property)`	String

Argument	Type	Details
this: `object`	Element	The feature to extract the property from.
`property`	String	The property to extract.

ee.ImageCollection.getVideoThumbURL

Get the URL of an animated thumbnail for this ImageCollection.

Returns a thumbnail URL, or undefined if a callback was specified.

Usage	Returns
`ImageCollection.getVideoThumbURL(params, `*`callback`*`)`	Object|String

Argument	Type	Details
this: `imagecollection`	ImageCollection	The ImageCollection instance.
`params`	Object	Parameters identical to ee.data.getMapId, plus, optionally: `dimensions` (a number or pair of numbers in format WIDTHxHEIGHT) Maximum dimensions of the thumbnail to render, in pixels. If only one number is passed, it is used as the maximum, and the other dimension is computed by proportional scaling. `region` (E,S,W,N or GeoJSON) Geospatial region of the image to render. By default, the whole image.

Argument	Type	Details
		`format` (string) Encoding format. Only 'gif' is accepted.
		`framesPerSecond` (number) Animation speed.
`callback`	*Function, optional*	*An optional callback which handles the resulting URL string. If not supplied, the call is made synchronously.*

ee.ImageCollection.iterate

Applies a user-supplied function to each element of a collection. The user-supplied function is given two arguments: the current element, and the value returned by the previous call to iterate() or the first argument, for the first iteration. The result is the value returned by the final call to the user-supplied function.

Returns the result of the Collection.iterate() call.

Usage	Returns
`ImageCollection.iterate(algorithm, `*`first`*`)`	ComputedObject

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`algorithm`	Function	The function to apply to each element. Must take two arguments: an element of the collection and the value from the previous iteration.
`first`	*Object, optional*	*The initial state.*

ee.ImageCollection.limit

Limit a collection to the specified number of elements, optionally sorting them by a specified property first.

Returns the limited collection.

Usage	Returns
`ImageCollection.limit(max, `*`property`*`, `*`ascending`*`)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`max`	Number	The number to limit the collection to.
`property`	*String, optional*	*The property to sort by, if sorting.*
`ascending`	*Boolean, optional*	*Whether to sort in ascending or descending order. The default is true (ascending).*

ee.ImageCollection.linkCollection

Links images in this collection to matching images from `imageCollection`.

For each source image in this collection, any specified bands or metadata will be added to the source image from the matching image found in

`imageCollection`. If bands or metadata are already present, they will be overwritten. If matching images are not found, any new or updated bands will be fully masked and any new or updated metadata will be null. The output footprint will be the same as the source image footprint.

Matches are determined if a source image and an image in `imageCollection` have a specific equivalent metadata property. If more than one collection image would match, the collection image selected is arbitrary. By default, images are matched on their 'system:index' metadata property.

This linking function is a convenience method for adding bands to target images based on a specified shared metadata property and is intended to support linking collections that apply different processing/product generation to the same source imagery. For more expressive linking known as

'joining', see https://developers.google.com/earth-engine/guides/joins_intro.

Returns the linked image collection.

Usage	Returns
`ImageCollection.linkCollection(imageCollection, `*`linkedBands`*`, `*`linkedProperties`*`, `*`matchPropertyName`*`)`	ImageCollection

Argument	Type	Details
this: `imagecollection`	ImageCollection	The ImageCollection instance.
`imageCollection`	ImageCollection	The image collection searched to find matches from this collection.
`linkedBands`	*List, optional*	*Optional list of band names to add or update from matching images.*
`linkedProperties`	*List, optional*	*Optional list of metadata properties to add or update from matching images.*

Argument	Type	Details
`matchPropertyName`	*String, optional*	*The metadata property name to use as a match criteria. Defaults to "system:index".*

ee.ImageCollection.load

Returns the image collection given its ID.

Usage	Returns
`ee.ImageCollection.load(id, version)`	ImageCollection

Argument	Type	Details
`id`	String	The asset ID of the image collection.
`version`	*Long, default: null*	*The version of the asset. -1 signifies the latest version.*

ee.ImageCollection.map

Maps an algorithm over a collection.

Returns the mapped collection.

Usage	Returns
`ImageCollection.map(algorithm, dropNulls)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`algorithm`	Function	The operation to map over the images or features of the collection. A JavaScript function that receives an image or features and returns one. The function is called only once and the result is captured as a description, so it cannot perform imperative operations or rely on external state.
`dropNulls`	*Boolean, optional*	*If true, the mapped algorithm is allowed to return nulls, and the elements for which it returns nulls will be dropped.*

ee.ImageCollection.max

Reduces an image collection by calculating the maximum value of each pixel across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.max()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.mean

Reduces an image collection by calculating the mean of all values at each pixel across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.mean()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.median

Reduces an image collection by calculating the median of all values at each pixel across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.median()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.merge

Merges two image collections into one. The result has all the images that were in either collection.

Usage	Returns
`ImageCollection.merge(collection2)`	ImageCollection

Argument	Type	Details
this: `collection1`	ImageCollection	The first collection to merge.
`collection2`	ImageCollection	The second collection to merge.

ee.ImageCollection.min

Reduces an image collection by calculating the minimum value of each pixel across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.min()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.mode

Reduces an image collection by calculating the most common value at each pixel across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.mode()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.mosaic

Composites all the images in a collection, using the mask.

Usage	Returns
`ImageCollection.mosaic()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The collection to mosaic.

ee.ImageCollection.or

Reduces an image collection by setting each pixel to 1 iff any of the non-masked values at that pixel are non-zero across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.or()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.product

Reduces an image collection by calculating the product of all values at each pixel across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.product()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.propertyNames

Returns the names of properties on this element.

Usage	Returns
`ImageCollection.propertyNames()`	List

Argument	Type	Details
this: `element`	Element	

ee.ImageCollection.qualityMosaic

Composites all the images in a collection, using a quality band as a per-pixel ordering function.

Usage	Returns
`ImageCollection.qualityMosaic(qualityBand)`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The collection to mosaic.
`qualityBand`	String	The name of the quality band in the collection.

ee.ImageCollection.randomColumn

Adds a column of deterministic pseudorandom numbers to a collection. The outputs are double-precision floating point numbers. When using the 'uniform' distribution (default), outputs are in the range of [0, 1). Using the 'normal' distribution, outputs have μ=0, **σ**=1, but have no explicit limits.

Usage	Returns
`ImageCollection.randomColumn(`*`columnName`*`, `*`seed`*`, `*`distribution`*`)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection to which to add a random column.
`columnName`	*String, default: "random"*	*The name of the column to add.*
`seed`	*Long, default: 0*	*A seed used when generating the random numbers.*
`distribution`	*String, default: "uniform"*	*The distribution type of random numbers to produce; one of 'uniform' or 'normal'.*

ee.ImageCollection.reduce

Applies a reducer across all of the images in a collection.

If the reducer has a single input, it will be applied separately to each band of the collection; otherwise it must have the same number of inputs as the collection has bands.

The reducer output names determine the names of the output bands: reducers with multiple inputs will use the output names directly, while reducers with a single input will prefix the output name with the input band name (e.g. '10_mean', '20_mean', etc.).

Usage	Returns
`ImageCollection.reduce(reducer, parallelScale)`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.
`reducer`	Reducer	The reducer to apply to the given collection.
`parallelScale`	*Float, default: 1*	*A scaling factor used to limit memory use; using a larger parallelScale (e.g. 2 or 4) may enable computations that run out of memory with the default.*

ee.ImageCollection.reduceColumns

Apply a reducer to each element of a collection, using the given selectors to determine the inputs.

Returns a dictionary of results, keyed with the output names.

Usage	Returns
`ImageCollection.reduceColumns(reducer, selectors, weightSelectors)`	Dictionary

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to aggregate over.
`reducer`	Reducer	The reducer to apply.
`selectors`	List	A selector for each input of the reducer.
`weightSelectors`	*List, default: null*	*A selector for each weighted input of the reducer.*

ee.ImageCollection.reduceToImage

Creates an image from a feature collection by applying a reducer over the selected properties of all the features that intersect each pixel.

Usage	Returns
`ImageCollection.reduceToImage(properties, reducer)`	Image

Argument	Type	Details
this: `collection`	FeatureCollection	Feature collection to intersect with each output pixel.
`properties`	List	Properties to select from each feature and pass into the reducer.
`reducer`	Reducer	A Reducer to combine the properties of each intersecting feature into a final result to store in the pixel.

ee.ImageCollection.remap

Remaps the value of a specific property in a collection. Takes two parallel lists and maps values found in one to values in the other. Any element with a value that is not specified in the first list is dropped from the output collection.

Usage	Returns
`ImageCollection.remap(lookupIn, lookupOut, columnName)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to be modified.
`lookupIn`	List	The input mapping values. Restricted to strings and integers.
`lookupOut`	List	The output mapping values. Must be the same size as lookupIn.
`columnName`	String	The name of the property to remap.

ee.ImageCollection.select

Select bands from each image in a collection.

Returns the image collection with selected bands.

Usage	Returns
`ImageCollection.select(selectors, names)`	ImageCollection

Argument	Type	Details
this: `imagecollection`	ImageCollection	The ImageCollection instance.
`selectors`	List	A list of names, regexes or numeric indices specifying the bands to select.
`names`	*List, optional*	*A list of new names for the output bands. Must match the number of bands selected.*

ee.ImageCollection.serialize

Returns the serialized representation of this object.

Usage	Returns
`ImageCollection.serialize(legacy)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.ImageCollection.set

Overrides one or more metadata properties of an Element.

Returns the element with the specified properties overridden.

Usage	Returns
`ImageCollection.set(var_args)`	Element

Argument	Type	Details
this: `element`	Element	The Element instance.
`var_args`	VarArgs	Either a dictionary of properties, or a vararg sequence of properties, e.g. key1, value1, key2, value2, ...

ee.ImageCollection.size

Returns the number of elements in the collection.

Usage	Returns
`ImageCollection.size()`	Integer

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to count.

ee.ImageCollection.sort

Sort a collection by the specified property.

Returns the sorted collection.

Usage	Returns
`ImageCollection.sort(property, `*`ascending`*`)`	Collection

Argument	Type	Details
this: `collection`	Collection	The Collection instance.
`property`	String	The property to sort by.
`ascending`	*Boolean, optional*	*Whether to sort in ascending or descending order. The default is true (ascending).*

ee.ImageCollection.style

Draw a vector collection for visualization using a simple style language.

Usage	Returns
`ImageCollection.style(`*`color`*`, `*`pointSize`*`, `*`pointShape`*`, `*`width`*`, `*`fillColor`*`, `*`styleProperty`*`, `*`neighborhood`*`, `*`lineType`*`)`	Image

Argument	Type	Details
this: `collection`	FeatureCollection	The collection to draw.
`color`	*String, default: "black"*	*A default color (CSS 3.0 color value e.g. 'FF0000' or 'red') to use for drawing the features. Supports opacity (e.g.: 'FF000088' for 50% transparent red).*
`pointSize`	*Integer, default: 3*	*The default size in pixels of the point markers.*
`pointShape`	*String, default: "circle"*	*The default shape of the marker to draw at each point location. One of: `circle`, `square`, `diamond`, `cross`, `plus`, `pentagram`, `hexagram`, `triangle`, `triangle_up`, `triangle_down`, `triangle_left`, `triangle_right`, `pentagon`, `hexagon`, `star5`, `star6`. This argument also supports the following Matlab marker abbreviations: `o`, `s`, `d`, `x`, `+`, `p`, `h`, `^`, `v`, `<`, `>`.*
`width`	*Float, default: 2*	*The default line width for lines and outlines for polygons and point shapes.*
`fillColor`	*String, default: null*	*The color for filling polygons and point shapes. Defaults to 'color' at 0.66 opacity.*
`styleProperty`	*String, default: null*	*A per-feature property expected to contain a dictionary. Values in the dictionary override any default values for that feature.*
`neighborhood`	*Integer, default: 5*	*If styleProperty is used and any feature has a pointSize or width larger than the defaults, tiling artifacts can occur. Specifies the maximum neighborhood (pointSize + width) needed for any feature.*
`lineType`	*String, default: "solid"*	*The default line style for lines and outlines of polygons and point shapes. Defaults to 'solid'. One of: solid, dotted, dashed.*

ee.ImageCollection.sum

Reduces an image collection by calculating the sum of all values at each pixel across the stack of all matching bands. Bands are matched by name.

Usage	Returns
`ImageCollection.sum()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The image collection to reduce.

ee.ImageCollection.toArray

Converts an image collection into an image of 2D arrays. At each pixel, the images that have valid (unmasked) values in all bands are laid out along the first axis of the array in the order they appear in the image collection.

The bands of each image are laid out along the second axis of the array, in the order the bands appear in that image. The array element type will be the union of the types of each band.

Usage	Returns
`ImageCollection.toArray()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	Image collection to convert to an array image. Bands must have scalar values, not array values.

ee.ImageCollection.toArrayPerBand

Concatenates multiple images into a single array image. The result will be masked if any input is masked.

Usage	Returns
`ImageCollection.toArrayPerBand(`*`axis`*`)`	Image

Argument	Type	Details
this: `collection`	ImageCollection	Images to concatenate. A separate concatenation is done per band, so all the images must have the same dimensionality and shape per band, except length along the concatenation axis.
`axis`	*Integer, default: 0*	*Axis to concatenate along; must be at least 0 and at most the minimum dimension of any band in the collection.*

ee.ImageCollection.toBands

Converts a collection to a single multi-band image containing all of the bands of every image in the collection. Output bands are named by prefixing the existing band names with the image id from which it came (e.g.: 'image1_band1'). Note: The maximum number of bands is 5000

Usage	Returns
`ImageCollection.toBands()`	Image

Argument	Type	Details
this: `collection`	ImageCollection	The input collection.

ee.ImageCollection.toDictionary

Extract properties from a feature as a dictionary.

Usage	Returns
`ImageCollection.toDictionary(`*`properties`*`)`	Dictionary

Argument	Type	Details
this: `element`	Element	The feature to extract the property from.
`properties`	*List, default: null*	*The list of properties to extract. Defaults to all non-system properties.*

ee.ImageCollection.toList

Returns the elements of a collection as a list.

Usage	Returns
`ImageCollection.toList(count, `*`offset`*`)`	List

Argument	Type	Details
this: `collection`	FeatureCollection	The input collection to fetch.
`count`	Integer	The maximum number of elements to fetch.
`offset`	*Integer, default: 0*	*The number of elements to discard from the start. If set, (offset + count) elements will be fetched and the first offset elements will be discarded.*

ee.ImageCollection.union

Merges all geometries in a given collection into one and returns a collection containing a single feature with only an ID of 'union_result' and a geometry.

Usage	Returns
`ImageCollection.union(`*`maxError`*`)`	FeatureCollection

Argument	Type	Details
this: `collection`	FeatureCollection	The collection being merged.
`maxError`	*ErrorMargin, default: null*	*The maximum error allowed when performing any necessary reprojections. If not specified, defaults to the error margin requested from the output.*

ee.Join.apply

Joins two collections.

Usage	Returns
`Join.apply(primary, secondary, condition)`	FeatureCollection

Argument	Type	Details
this: `join`	Join	The join to apply; determines how the the results are constructed.
`primary`	FeatureCollection	The primary collection.
`secondary`	FeatureCollection	The secondary collection.
`condition`	Filter	The join condition used to select the matches from the two collections.

ee.Join.inner

Returns a join that pairs elements from the primary collection with matching elements from the secondary collection. Each result has a 'primary' property that contains the element from the primary collection, and a 'secondary' property containing the matching element from the secondary collection. If measureKey is specified, the join measure is also attached to the object as a property.

Usage	Returns
`ee.Join.inner(`*`primaryKey`*`, `*`secondaryKey`*`, `*`measureKey`*`)`	Join

Argument	Type	Details
`primaryKey`	*String, default: "primary"*	*The property name used to save the primary match.*
`secondaryKey`	*String, default: "secondary"*	*The property name used to save the secondary match.*

Argument	Type	Details
`measureKey`	*String, default: null*	*An optional property name used to save the measure of the join condition.*

ee.Join.inverted

Returns a join that produces the elements of the primary collection that match no elements of the secondary collection. No properties are added to the results.

Usage	Returns
`ee.Join.inverted()`	Join

No arguments.

ee.Join.saveAll

Returns a join that pairs each element from the first collection with a group of matching elements from the second collection. The list of matches is added to each result as an additional property. If measureKey is specified, each match has the value of its join measure attached. Join measures are produced when withinDistance or maxDifference filters are used as the join condition.

Usage	Returns
`ee.Join.saveAll(matchesKey, ordering, ascending, measureKey, outer)`	Join

Argument	Type	Details
`matchesKey`	String	The property name used to save the matches list.
`ordering`	*String, default: null*	*The property on which to sort the matches list.*
`ascending`	*Boolean, default: true*	*Whether the ordering is ascending.*
`measureKey`	*String, default: null*	*An optional property name used to save the measure of the join condition on each match.*
`outer`	*Boolean, default: false*	*If true, primary rows without matches will be included in the result.*

ee.Join.saveBest

Returns a join that pairs each element from the first collection with a matching element from the second collection. The match with the best join measure is added to each result as an additional property. Join measures are produced when withinDistance or maxDifference filters are used as the join condition.

Usage	Returns
`ee.Join.saveBest(matchKey, measureKey, outer)`	Join

Argument	Type	Details
`matchKey`	String	The key used to save the match.
`measureKey`	String	The key used to save the measure of the join condition on the match.
`outer`	*Boolean, default: false*	*If true, primary rows without matches will be included in the result.*

ee.Join.saveFirst

Returns a join that pairs each element from the first collection with a matching element from the second collection. The first match is added to the result as an additional property.

Usage	Returns
`ee.Join.saveFirst(matchKey, ordering, ascending, measureKey, outer)`	Join

Argument	Type	Details
`matchKey`	String	The property name used to save the match.
`ordering`	*String, default: null*	*The property on which to sort the matches before selecting the first.*
`ascending`	*Boolean, default: true*	*Whether the ordering is ascending.*
`measureKey`	*String, default: null*	*An optional property name used to save the measure of the join condition on the match.*
`outer`	*Boolean, default: false*	*If true, primary rows without matches will be included in the result.*

ee.Join.simple

Returns a join that produces the elements of the primary collection that match any element of the secondary collection. No properties are added to the results.

Usage	Returns
`ee.Join.simple()`	Join

No arguments.

ee.Kernel.add

Adds two kernels (pointwise), after aligning their centers.

Usage	Returns
`Kernel.add(kernel2, normalize)`	Kernel

Argument	Type	Details
this: `kernel1`	Kernel	The first kernel.
`kernel2`	Kernel	The second kernel.
`normalize`	*Boolean, default: false*	*Normalize the kernel.*

ee.Kernel.chebyshev

Generates a distance kernel based on Chebyshev distance (greatest distance along any dimension).

Usage	Returns
`ee.Kernel.chebyshev(radius, units, normalize, magnitude)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.circle

Generates a circle-shaped boolean kernel.

Usage	Returns
`ee.Kernel.circle(radius, units, normalize, magnitude)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: true*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.compass

Generates a 3x3 Prewitt's Compass edge-detection kernel.

Usage	Returns
`ee.Kernel.compass(magnitude, normalize)`	Kernel

Argument	Type	Details
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

ee.Kernel.cross

Generates a cross-shaped boolean kernel.

Usage	Returns
`ee.Kernel.cross(radius, units, normalize, magnitude)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: true*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.diamond

Generates a diamond-shaped boolean kernel.

Usage	Returns
`ee.Kernel.diamond(radius, `*`units`*`, `*`normalize`*`, `*`magnitude`*`)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: true*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.euclidean

Generates a distance kernel based on Euclidean (straight-line) distance.

Usage	Returns
`ee.Kernel.euclidean(radius, `*`units`*`, `*`normalize`*`, `*`magnitude`*`)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.

Argument	Type	Details
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.fixed

Creates a Kernel.

Usage	Returns
`ee.Kernel.fixed(`*`width`*`, `*`height`*`, weights, `*`x`*`, `*`y`*`, `*`normalize`*`)`	Kernel

Argument	Type	Details
`width`	*Integer, default: -1*	*The width of the kernel in pixels.*
`height`	*Integer, default: -1*	*The height of the kernel in pixels.*
`weights`	List	A 2-D list of [height] x [width] values to use as the weights of the kernel.
`x`	*Integer, default: -1*	*The location of the focus, as an offset from the left.*
`y`	*Integer, default: -1*	*The location of the focus, as an offset from the top.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

ee.Kernel.gaussian

Generates a Gaussian kernel from a sampled continuous Gaussian.

Usage	Returns
`ee.Kernel.gaussian(radius, `*`sigma`*`, `*`units`*`, `*`normalize`*`, `*`magnitude`*`)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.

Argument	Type	Details
`sigma`	*Float, default: 1*	*Standard deviation of the Gaussian function (same units as radius).*
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: true*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.inverse

Returns a kernel which has each of its weights multiplicatively inverted. Weights with a value of zero are not inverted and remain zero.

Usage	Returns
`Kernel.inverse()`	Kernel

Argument	Type	Details
this: `kernel`	Kernel	The kernel to have its entries inverted.

ee.Kernel.kirsch

Generates a 3x3 Kirsch's Compass edge-detection kernel.

Usage	Returns
`ee.Kernel.kirsch(`*`magnitude`*`, `*`normalize`*`)`	Kernel

Argument	Type	Details
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

ee.Kernel.laplacian4

Generates a 3x3 Laplacian-4 edge-detection kernel.

Usage	Returns
`ee.Kernel.laplacian4(`*`magnitude`*`, `*`normalize`*`)`	Kernel

Argument	Type	Details
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

ee.Kernel.laplacian8

Generates a 3x3 Laplacian-8 edge-detection kernel.

Usage	Returns
`ee.Kernel.laplacian8(`*`magnitude`*`, `*`normalize`*`)`	Kernel

Argument	Type	Details
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

ee.Kernel.manhattan

Generates a distance kernel based on rectilinear (city-block) distance.

Usage	Returns
`ee.Kernel.manhattan(radius, `*`units`*`, `*`normalize`*`, `*`magnitude`*`)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

Argument	Type	Details
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.octagon

Generates an octagon-shaped boolean kernel.

Usage	Returns
`ee.Kernel.octagon(radius, units, normalize, magnitude)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: true*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.plus

Generates a plus-shaped boolean kernel.

Usage	Returns
`ee.Kernel.plus(radius, units, normalize, magnitude)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: true*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.prewitt

Generates a 3x3 Prewitt edge-detection kernel.

Usage	Returns
`ee.Kernel.prewitt(`*`magnitude`*`, `*`normalize`*`)`	Kernel

Argument	Type	Details
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

ee.Kernel.rectangle

Generates a rectangular-shaped kernel.

Usage	Returns
`ee.Kernel.rectangle(xRadius, yRadius, `*`units`*`, `*`normalize`*`, `*`magnitude`*`)`	Kernel

Argument	Type	Details
`xRadius`	Float	The horizontal radius of the kernel to generate.
`yRadius`	Float	The vertical radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ("pixels" or "meters"). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: true*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.Kernel.roberts

Generates a 2x2 Roberts edge-detection kernel.

Usage	Returns
`ee.Kernel.roberts(`*`magnitude`*`, `*`normalize`*`)`	Kernel

Argument	Type	Details
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

ee.Kernel.rotate

Creates a Kernel.

Usage	Returns
`Kernel.rotate(rotations)`	Kernel

Argument	Type	Details
this: `kernel`	Kernel	The kernel to be rotated.
`rotations`	Integer	Number of 90 deg. rotations to make (negative numbers rotate counterclockwise).

ee.Kernel.sobel

Generates a 3x3 Sobel edge-detection kernel.

Usage	Returns
`ee.Kernel.sobel(`*`magnitude`*`, `*`normalize`*`)`	Kernel

Argument	Type	Details
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*
`normalize`	*Boolean, default: false*	*Normalize the kernel values to sum to 1.*

ee.Kernel.square

Generates a square-shaped boolean kernel.

Usage	Returns
`ee.Kernel.square(radius, units, normalize, magnitude)`	Kernel

Argument	Type	Details
`radius`	Float	The radius of the kernel to generate.
`units`	*String, default: "pixels"*	*The system of measurement for the kernel ('pixels' or 'meters'). If the kernel is specified in meters, it will resize when the zoom-level is changed.*
`normalize`	*Boolean, default: true*	*Normalize the kernel values to sum to 1.*
`magnitude`	*Float, default: 1*	*Scale each value by this amount.*

ee.List

Constructs a new list.

Usage	Returns
`ee.List(list)`	List

Argument	Type	Details
`list`	List	A list or a computed object.

ee.List.add

Appends the element to the end of list.

Usage	Returns
`List.add(element)`	List

Argument	Type	Details
this: `list`	List	
`element`	Object	

ee.List.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`List.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.List.cat

Concatenates the contents of other onto list.

Usage	Returns
`List.cat(other)`	List

Argument	Type	Details
this: `list`	List	
`other`	List	

ee.List.contains

Returns true if list contains element.

Usage	Returns
`List.contains(element)`	Boolean

Argument	Type	Details
this: `list`	List	
`element`	Object	

ee.List.containsAll

Returns true if list contains all of the elements of other, regardless of order.

Usage	Returns
`List.containsAll(other)`	Boolean

Argument	Type	Details
this: `list`	List	
`other`	List	

ee.List.distinct

Returns a copy of list without duplicate elements.

Usage	Returns
`List.distinct()`	List

Argument	Type	Details
this: `list`	List	

ee.List.equals

Returns true if list contains the same elements as other, in the same order.

Usage	Returns
`List.equals(other)`	Boolean

Argument	Type	Details
this: `list`	List	
`other`	List	

ee.List.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`List.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.List.filter

Filters a list to only the elements that match the given filter. To filter list items that aren't images or features, test a property named 'item', e.g.: ee.Filter.gt('item', 3)

Usage	Returns
`List.filter(filter)`	List

Argument	Type	Details
this: `list`	List	

Argument	Type	Details
`filter`	Filter	

ee.List.flatten

Flattens any sublists into a single list.

Usage	Returns
`List.flatten()`	List

Argument	Type	Details
this: `list`	List	

ee.List.frequency

Returns the number of elements in list equal to element.

Usage	Returns
`List.frequency(element)`	Integer

Argument	Type	Details
this: `list`	List	
`element`	Object	

ee.List.get

Returns the element at the specified position in list. A negative index counts backwards from the end of the list.

Usage	Returns
`List.get(index)`	Object

Argument	Type	Details
this: `list`	List	
`index`	Integer	

ee.List.getArray

Returns the array at the specified position in list. A negative index counts backwards from the end of the list. If the value is not a array, an error will occur.

Usage	Returns
`List.getArray(index)`	Array

Argument	Type	Details
this: `list`	List	
`index`	Integer	

ee.List.getGeometry

Returns the geometry at the specified position in list. A negative index counts backwards from the end of the list. If the value is not a geometry, an error will occur.

Usage	Returns
`List.getGeometry(index)`	Geometry

Argument	Type	Details
this: `list`	List	
`index`	Integer	

ee.List.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`List.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.List.getNumber

Returns the number at the specified position in list. A negative index counts backwards from the end of the list. If the value is not a number, an error will occur.

Usage	Returns
`List.getNumber(index)`	Number

Argument	Type	Details
this: `list`	List	
`index`	Integer	

ee.List.getString

Returns the string at the specified position in list. A negative index counts backwards from the end of the list. If the value is not a string, an error will occur.

Usage	Returns
`List.getString(index)`	String

Argument	Type	Details
this: `list`	List	
`index`	Integer	

ee.List.indexOf

Returns the position of the first occurrence of target in list, or -1 if list does not contain target.

Usage	Returns
`List.indexOf(element)`	Integer

Argument	Type	Details
this: `list`	List	
`element`	Object	

ee.List.indexOfSublist

Returns the starting position of the first occurrence of target within list, or -1 if there is no such occurrence.

Usage	Returns
`List.indexOfSublist(target)`	Integer

Argument	Type	Details
this: `list`	List	
`target`	List	

ee.List.insert

Inserts element at the specified position in list. A negative index counts backwards from the end of the list.

Usage	Returns
`List.insert(index, element)`	List

Argument	Type	Details
this: `list`	List	
`index`	Integer	
`element`	Object	

ee.List.iterate

Iterate an algorithm over a list. The algorithm is expected to take two objects, the current list item, and the result from the previous iteration or the value of first for the first iteration.

Usage	Returns
`List.iterate(function, first)`	Object

Argument	Type	Details
this: `list`	List	
`function`	Algorithm	
`first`	Object	

ee.List.join

Returns a string containing the elements of the list joined together with the specified separator between elements.

Note: The string form of list elements which are not strings, numbers, or booleans is currently not well-defined and subject to change.

Usage	Returns
`List.join(`*`separator`*`)`	String

Argument	Type	Details
this: `list`	List	
`separator`	*String, default: ""*	

ee.List.lastIndexOfSubList

Returns the starting position of the last occurrence of target within list, or -1 if there is no such occurrence.

Usage	Returns
`List.lastIndexOfSubList(target)`	Integer

Argument	Type	Details
this: `list`	List	
`target`	List	

ee.List.length

Returns the number of elements in list.

Usage	Returns
`List.length()`	Integer

Argument	Type	Details
this: `list`	List	

ee.List.map

Map an algorithm over a list. The algorithm is expected to take an Object and return an Object.

Usage	Returns
`List.map(baseAlgorithm, dropNulls)`	List

Argument	Type	Details
this: `list`	List	
`baseAlgorithm`	Algorithm	
`dropNulls`	*Boolean, default: false*	*If true, the mapped algorithm is allowed to return nulls, and the elements for which it returns nulls will be dropped.*

ee.List.reduce

Apply a reducer to a list. If the reducer takes more than 1 input, then each element in the list is assumed to be a list of inputs. If the reducer returns a single output, it is returned directly, otherwise returns a dictionary containing the named reducer outputs.

Usage	Returns
`List.reduce(reducer)`	Object

Argument	Type	Details
this: `list`	List	
`reducer`	Reducer	

ee.List.remove

Removes the first occurrence of the specified element from list, if it is present.

Usage	Returns
`List.remove(element)`	List

Argument	Type	Details
this: `list`	List	
`element`	Object	

ee.List.removeAll

Removes from list all of the elements that are contained in other list.

Usage	Returns
`List.removeAll(other)`	List

Argument	Type	Details
this: `list`	List	
`other`	List	

ee.List.repeat

Returns a new list containing value repeated count times.

Usage	Returns
`ee.List.repeat(value, count)`	List

Argument	Type	Details
`value`	Object	
`count`	Integer	

ee.List.replace

Replaces the first occurrence of oldVal in list with newVal.

Usage	Returns
`List.replace(oldval, newval)`	List

Argument	Type	Details
this: `list`	List	
`oldval`	Object	
`newval`	Object	

ee.List.replaceAll

Replaces all occurrences of oldVal in list with newVal.

Usage	Returns
`List.replaceAll(oldval, newval)`	List

Argument	Type	Details
this: `list`	List	
`oldval`	Object	
`newval`	Object	

ee.List.reverse

Reverses the order of the elements in list.

Usage	Returns
`List.reverse()`	List

Argument	Type	Details
this: `list`	List	

ee.List.rotate

Rotates the elements of the list by the specified distance.

Usage	Returns
`List.rotate(distance)`	List

Argument	Type	Details
this: `list`	List	
`distance`	Integer	

ee.List.sequence

Generate a sequence of numbers from start to end (inclusive) in increments of step, or in count equally-spaced increments. If end is not specified it is computed from start + step * count, so at least one of end or count must be specified.

Usage	Returns
`ee.List.sequence(start, end, step, count)`	List

Argument	Type	Details
`start`	Number	
`end`	*Number, default: null*	
`step`	*Number, default: 1*	
`count`	*Integer, default: null*	

ee.List.serialize

Returns the serialized representation of this object.

Usage	Returns
`List.serialize(legacy)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.List.set

Replaces the value at the specified position in list with element. A negative index counts backwards from the end of the list.

Usage	Returns
`List.set(index, element)`	List

Argument	Type	Details
this: `list`	List	
`index`	Integer	
`element`	Object	

ee.List.shuffle

Randomly permute the specified list. Note that the permutation order will always be the same for any given seed, unless the value for seed is 'false'.

Usage	Returns
`List.shuffle(`*seed*`)`	List

Argument	Type	Details
this: `list`	List	
`seed`	*Object, default: null*	*A long integer to use as a seed for the randomization. If the boolean value of 'false' is passed, then a completely random and unreproducible order will be generated.*

ee.List.size

Returns the number of elements in list.

Usage	Returns
`List.size()`	Integer

Argument	Type	Details
this: `list`	List	

ee.List.slice

Returns a portion of list between the start index, inclusive, and end index, exclusive. Negative values for start or end count backwards from the end of the list. Values greater than the size of the list are legal but are truncated

to the size of list.

Usage	Returns
`List.slice(start, end, step)`	List

Argument	Type	Details
this: `list`	List	
`start`	Integer	
`end`	*Integer, default: null*	
`step`	*Integer, default: null*	

ee.List.sort

Sorts the list into ascending order. If the 'keys' argument is provided, then it is sorted first, and the elements of 'list' are placed in the same order.

Usage	Returns
`List.sort(keys)`	List

Argument	Type	Details
this: `list`	List	The list to sort.
`keys`	*List, default: null*	*Optional keys to sort by. If 'keys' is provided, it must have the same length as 'list'.*

ee.List.splice

Starting at the start index, removes count elements from list and insert the contents of other at that location. If start is negative, it counts backwards from the end of the list.

Usage	Returns
`List.splice(start, count, other)`	List

Argument	Type	Details
this: `list`	List	
`start`	Integer	
`count`	Integer	
`other`	*List, default: null*	

ee.List.swap

Swaps the elements at the specified positions. A negative position counts backwards from the end of the list.

Usage	Returns
`List.swap(pos1, pos2)`	List

Argument	Type	Details
this: `list`	List	
`pos1`	Integer	
`pos2`	Integer	

ee.List.unzip

Transposes a list of lists, extracting the first element of each inner list into one list the second elements into another, etc., up to the length of the shortest inner list. The remaining items are discarded. The result is a list of lists.

Usage	Returns
`List.unzip()`	List

Argument	Type	Details
this: `list`	List	

ee.List.zip

Pairs the elements of two lists to create a list of two-element lists. When the input lists are of different sizes, the final list has the same size as the shortest one.

Usage	Returns
`List.zip(other)`	List

Argument	Type	Details
this: `list`	List	
`other`	List	

ee.Model.fromAiPlatformPredictor

Returns an ee.Model from a description of an AI Platform prediction model. (See https://cloud.google.com/ml-engine/).

Usage	Returns
`ee.Model.fromAiPlatformPredictor(`*`projectName`*`, `*`projectId`*`, `*`modelName`*`, `*`version`*`, `*`region`*`, `*`inputProperties`*`, `*`inputTypeOverride`*`, `*`inputShapes`*`, `*`proj`*`, `*`fixInputProj`*`, `*`inputTileSize`*`, `*`inputOverlapSize`*`, `*`outputTileSize`*`, `*`outputBands`*`, `*`outputProperties`*`, `*`outputMultiplier`*`)`	Model

Argument	Type	Details
`projectName`	*Object, default: null*	*The Google Cloud project that owns the model. Deprecated: use "projectId" instead.*
`projectId`	*String, default: null*	*The ID of the Google Cloud project that owns the model.*
`modelName`	*String, default: null*	*The name of the model.*
`version`	*String, default: null*	*The model version. Defaults to the AI Platform default model version.*
`region`	*String, default: null*	*The model deployment region. Defaults to "us-central1".*
`inputProperties`	*List, default: null*	*Properties passed with each prediction instance. Image predictions are tiled, so these properties will be replicated into each image tile instance. Defaults to no properties.*
`inputTypeOverride`	*Dictionary, default: null*	*Types to which model inputs will be coerced if specified. Both Image bands and Image/Feature properties are valid.*

Argument	Type	Details
`inputShapes`	*Dictionary, default: null*	*The fixed shape of input array bands. For each array band not specified, the fixed array shape will be automatically deduced from a non-masked pixel.*
`proj`	*Projection, default: null*	*The input projection at which to sample all bands. Defaults to the default projection of an image's first band.*
`fixInputProj`	*Boolean, default: null*	*If true, pixels will be sampled in a fixed projection specified by 'proj'. The output projection is used otherwise. Defaults to false.*
`inputTileSize`	*List, default: null*	*Rectangular dimensions of pixel tiles passed in to prediction instances. Required for image predictions.*
`inputOverlapSize`	*List, default: null*	*Amount of adjacent-tile overlap in X/Y along each edge of pixel tiles passed in to prediction instances. Defaults to [0, 0].*
`outputTileSize`	*List, default: null*	*Rectangular dimensions of pixel tiles returned from AI Platform. Defaults to the value in 'inputTileSize'.*
`outputBands`	*Dictionary, default: null*	*A map from output band names to a dictionary of output band info. Valid band info fields are 'type' and 'dimensions'. 'type' should be a ee.PixelType describing the output band, and 'dimensions' is an optional integer with the number of dimensions in that band e.g.: "outputBands: {'p': {'type': ee.PixelType.int8(), 'dimensions': 1}}". Required for image predictions.*
`outputProperties`	*Dictionary, default: null*	*A map from output property names to a dictionary of output property info. Valid property info fields are 'type' and 'dimensions'. 'type' should be a ee.PixelType describing the output property, and 'dimensions' is an optional integer with the number of dimensions for that property if it is an array e.g.: "outputBands: {'p': {'type': ee.PixelType.int8(), 'dimensions': 1}}". Required for predictions from FeatureCollections.*
`outputMultiplier`	*Float, default: null*	*An approximation to the increase in data volume for the model outputs over the model inputs. If specified this must be >= 1. This is only needed if the model produces more data than it consumes, e.g. a model that takes 5 bands and produces 10 outputs per pixel.*

ee.Model.fromVertexAi

Returns an ee.Model from a description of a Vertex AI model endpoint. (See https://cloud.google.com/vertex-ai).

Warning: This method is in public preview and may undergo breaking changes.

Usage	Returns
`ee.Model.fromVertexAi(`*`endpoint`*`, `*`inputProperties`*`, `*`inputTypeOverride`*`, `*`inputShapes`*`, `*`proj`*`, `*`fixInputProj`*`, `*`inputTileSize`*`, `*`inputOverlapSize`*`, `*`outputTileSize`*`, `*`outputBands`*`, `*`outputProperties`*`, `*`outputMultiplier`*`, `*`maxPayloadBytes`*`, `*`payloadFormat`*`)`	Model

Argument	Type	Details
`endpoint`	*String, default: null*	*The endpoint name for predictions.*
`inputProperties`	*List, default: null*	*Properties passed with each prediction instance. Image predictions are tiled, so these properties will be replicated into each image tile instance. Defaults to no properties.*
`inputTypeOverride`	*Dictionary, default: null*	*Types to which model inputs will be coerced if specified. Both Image bands and Image/Feature properties are valid.*
`inputShapes`	*Dictionary, default: null*	*The fixed shape of input array bands. For each array band not specified, the fixed array shape will be automatically deduced from a non-masked pixel.*
`proj`	*Projection, default: null*	*The input projection at which to sample all bands. Defaults to the default projection of an image's first band.*
`fixInputProj`	*Boolean, default: null*	*If true, pixels will be sampled in a fixed projection specified by 'proj'. The output projection is used otherwise. Defaults to false.*
`inputTileSize`	*List, default: null*	*Rectangular dimensions of pixel tiles passed in to prediction instances. Required for image predictions.*
`inputOverlapSize`	*List, default: null*	*Amount of adjacent-tile overlap in X/Y along each edge of pixel tiles passed in to prediction instances. Defaults to [0, 0].*
`outputTileSize`	*List, default: null*	*Rectangular dimensions of pixel tiles returned from AI Platform. Defaults to the value in 'inputTileSize'.*
`outputBands`	*Dictionary, default: null*	*A map from output band names to a dictionary of output band info. Valid band info fields are 'type' and 'dimensions'. 'type' should be a ee.PixelType describing the output band, and 'dimensions' is an optional integer with the number of dimensions in that band e.g.: "outputBands: {'p': {'type': ee.PixelType.int8(), 'dimensions': 1}}". Required for image predictions.*
`outputProperties`	*Dictionary, default: null*	*A map from output property names to a dictionary of output property info. Valid property info fields are 'type' and 'dimensions'. 'type' should be a ee.PixelType describing the output property, and 'dimensions' is an optional integer with the number of dimensions for that property if it is an array e.g.: "outputBands: {'p': {'type': ee.PixelType.int8(), 'dimensions': 1}}". Required for predictions from FeatureCollections.*
`outputMultiplier`	*Float, default: null*	*An approximation to the increase in data volume for the model outputs over the model inputs. If specified this must be >= 1. This is only needed if the model produces more data than it consumes, e.g. a model that takes 5 bands and produces 10 outputs per pixel.*
`maxPayloadBytes`	*Long, default: null*	*The prediction payload size limit in bytes. Defaults to 1.5MB (1500000 bytes)*
`payloadFormat`	*String, default: null*	*The payload format of entries in prediction requests and responses. One of: ['SERIALIZED_TF_TENSORS, 'RAW_JSON', 'ND_ARRAYS']. Defaults to 'SERIALIZED_TF_TENSORS'.*

ee.Model.predictImage

Make predictions from pixel tiles of an image. The predictions are merged as bands with the input image.

The model will receive 0s in place of masked pixels. The masks of predicted output bands are the minimum of the masks of the inputs.

Usage	Returns
`Model.predictImage(image)`	Image

Argument	Type	Details
this: `model`	Model	
`image`	Image	The input image.

ee.Model.predictProperties

Make predictions for each feature in a collection. Predicted properties are merged with the properties of the input feature.

Usage	Returns
`Model.predictProperties(collection)`	FeatureCollection

Argument	Type	Details
this: `model`	Model	
`collection`	FeatureCollection	The input collection.

ee.Number

Constructs a new Number.

Usage	Returns
`ee.Number(number)`	Number

Argument	Type	Details
`number`	Number|Object	A number or a computed object.

ee.Number.abs

Computes the absolute value of the input.

Usage	Returns
`Number.abs()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.acos

Computes the arc cosine in radians of the input.

Usage	Returns
`Number.acos()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.add

Adds the first value to the second.

Usage	Returns
`Number.add(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.and

Returns 1 if and only if both values are non-zero.

Usage	Returns
`Number.and(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`Number.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.

Argument	Type	Details
`func`	Function	The function to call.
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.Number.asin

Computes the arc sine in radians of the input.

Usage	Returns
`Number.asin()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.atan

Computes the arc tangent in radians of the input.

Usage	Returns
`Number.atan()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.atan2

Calculates the angle formed by the 2D vector [x, y].

Usage	Returns
`Number.atan2(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.bitCount

Calculates the number of one-bits in the 64-bit two's complement binary representation of the input.

Usage	Returns
`Number.bitCount()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.bitwiseAnd

Calculates the bitwise AND of the input values.

Usage	Returns
`Number.bitwiseAnd(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.bitwiseNot

Calculates the bitwise NOT of the input, in the smallest signed integer type that can hold the input.

Usage	Returns
`Number.bitwiseNot()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.bitwiseOr

Calculates the bitwise OR of the input values.

Usage	Returns
`Number.bitwiseOr(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.bitwiseXor

Calculates the bitwise XOR of the input values.

Usage	Returns
`Number.bitwiseXor(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.byte

Casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Number.byte()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.cbrt

Computes the cubic root of the input.

Usage	Returns
`Number.cbrt()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.ceil

Computes the smallest integer greater than or equal to the input.

Usage	Returns
`Number.ceil()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.clamp

Clamps the value to lie within the range of min to max.

Usage	Returns
`Number.clamp(min, max)`	Number

Argument	Type	Details
this: `number`	Number	
`min`	Float	
`max`	Float	

ee.Number.cos

Computes the cosine of the input in radians.

Usage	Returns
`Number.cos()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.cosh

Computes the hyperbolic cosine of the input.

Usage	Returns
`Number.cosh()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.digamma

Computes the digamma function of the input.

Usage	Returns
`Number.digamma()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.divide

Divides the first value by the second, returning 0 for division by 0.

Usage	Returns
`Number.divide(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.double

Casts the input value to a 64-bit float.

Usage	Returns
`Number.double()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.eq

Returns 1 if and only if the first value is equal to the second.

Usage	Returns
`Number.eq(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.erf

Computes the error function of the input.

Usage	Returns
`Number.erf()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.erfInv

Computes the inverse error function of the input.

Usage	Returns
`Number.erfInv()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.erfc

Computes the complementary error function of the input.

Usage	Returns
`Number.erfc()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.erfcInv

Computes the inverse complementary error function of the input.

Usage	Returns
`Number.erfcInv()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`Number.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.Number.exp

Computes the Euler's number e raised to the power of the input.

Usage	Returns
`Number.exp()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.expression

Computes a numeric expression.

Usage	Returns
`ee.Number.expression(expression, vars)`	Number

Argument	Type	Details
`expression`	String	A mathematical expression string to be evaluated. In addition to the standard arithmetic, boolean and relational operators, expressions also support any function in Number, the '.' operator to extract child elements from the 'vars' dictionary, and mathematical constants Math.PI and Math.E
`vars`	*Dictionary, default: null*	*A dictionary of named values that can be used in the expression.*

ee.Number.first

Selects the value of the first value.

Usage	Returns
`Number.first(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.firstNonZero

Selects the first value if it is non-zero, and the second value otherwise.

Usage	Returns
`Number.firstNonZero(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.float

Casts the input value to a 32-bit float.

Usage	Returns
`Number.float()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.floor

Computes the largest integer less than or equal to the input.

Usage	Returns
`Number.floor()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.format

Convert a number to a string using printf-style formatting.

Usage	Returns
`Number.format(`*`pattern`*`)`	String

Argument	Type	Details
this: `number`	Number	The number to convert to a string.
`pattern`	*String, default: "%s"*	*A printf-style format string. For example, '%.2f' produces numbers formatted like '3.14', and '%05d' produces numbers formatted like '00042'. The format string must satisfy the following criteria:* *1. Zero or more prefix characters.* *2. Exactly one '%'.* *3. Zero or more modifier characters in the set [#-+ 0,(.\d].* *4. Exactly one conversion character in the set [sdoxXeEfgGaA].* *5. Zero or more suffix characters.* *For more about format strings, see https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html*

ee.Number.gamma

Computes the gamma function of the input.

Usage	Returns
`Number.gamma()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.gammainc

Calculates the regularized lower incomplete Gamma function γ(x,a).

Usage	Returns
`Number.gammainc(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`Number.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.Number.gt

Returns 1 if and only if the first value is greater than the second.

Usage	Returns
`Number.gt(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.gte

Returns 1 if and only if the first value is greater than or equal to the second.

Usage	Returns
`Number.gte(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.hypot

Calculates the magnitude of the 2D vector [x, y].

Usage	Returns
`Number.hypot(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.int

Casts the input value to a signed 32-bit integer.

Usage	Returns
`Number.int()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.int16

Casts the input value to a signed 16-bit integer.

Usage	Returns
`Number.int16()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.int32

Casts the input value to a signed 32-bit integer.

Usage	Returns
`Number.int32()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.int64

Casts the input value to a signed 64-bit integer.

Usage	Returns
`Number.int64()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.int8

Casts the input value to a signed 8-bit integer.

Usage	Returns
`Number.int8()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.lanczos

Computes the Lanczos approximation of the input.

Usage	Returns
`Number.lanczos()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.leftShift

Calculates the left shift of v1 by v2 bits.

Usage	Returns
`Number.leftShift(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.log

Computes the natural logarithm of the input.

Usage	Returns
`Number.log()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.log10

Computes the base-10 logarithm of the input.

Usage	Returns
`Number.log10()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.long

Casts the input value to a signed 64-bit integer.

Usage	Returns
`Number.long()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.lt

Returns 1 if and only if the first value is less than the second.

Usage	Returns
`Number.lt(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.lte

Returns 1 if and only if the first value is less than or equal to the second.

Usage	Returns
`Number.lte(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.max

Selects the maximum of the first and second values.

Usage	Returns
`Number.max(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.min

Selects the minimum of the first and second values.

Usage	Returns
`Number.min(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.mod

Calculates the remainder of the first value divided by the second.

Usage	Returns
`Number.mod(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.multiply

Multiplies the first value by the second.

Usage	Returns
`Number.multiply(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.neq

Returns 1 if and only if the first value is not equal to the second.

Usage	Returns
`Number.neq(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.not

Returns 0 if the input is non-zero, and 1 otherwise.

Usage	Returns
`Number.not()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.or

Returns 1 if and only if either input value is non-zero.

Usage	Returns
`Number.or(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.parse

Convert a string to a number.

Usage	Returns
`ee.Number.parse(input, radix)`	Number

Argument	Type	Details
`input`	String	The string to convert to a number.
`radix`	*Integer, default: 10*	*An integer representing the base number system from which to convert. If input is not an integer, radix must equal 10 or not be specified.*

ee.Number.pow

Raises the first value to the power of the second.

Usage	Returns
`Number.pow(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.rightShift

Calculates the signed right shift of v1 by v2 bits.

Usage	Returns
`Number.rightShift(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.round

Computes the integer nearest to the input.

Usage	Returns
`Number.round()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.serialize

Returns the serialized representation of this object.

Usage	Returns
`Number.serialize(`*`legacy`*`)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.Number.short

Casts the input value to a signed 16-bit integer.

Usage	Returns
`Number.short()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.signum

Computes the signum function (sign) of the input; zero if the input is zero, 1 if the input is greater than zero, -1 if the input is less than zero.

Usage	Returns
`Number.signum()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.sin

Computes the sine of the input in radians.

Usage	Returns
`Number.sin()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.sinh

Computes the hyperbolic sine of the input.

Usage	Returns
`Number.sinh()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.sqrt

Computes the square root of the input.

Usage	Returns
`Number.sqrt()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.subtract

Subtracts the second value from the first.

Usage	Returns
`Number.subtract(right)`	Number

Argument	Type	Details
this: `left`	Number	The left-hand value.
`right`	Number	The right-hand value.

ee.Number.tan

Computes the tangent of the input in radians.

Usage	Returns
`Number.tan()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.tanh

Computes the hyperbolic tangent of the input.

Usage	Returns
`Number.tanh()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toByte

Casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Number.toByte()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toDouble

Casts the input value to a 64-bit float.

Usage	Returns
`Number.toDouble()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toFloat

Casts the input value to a 32-bit float.

Usage	Returns
`Number.toFloat()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toInt

Casts the input value to a signed 32-bit integer.

Usage	Returns
`Number.toInt()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toInt16

Casts the input value to a signed 16-bit integer.

Usage	Returns
`Number.toInt16()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toInt32

Casts the input value to a signed 32-bit integer.

Usage	Returns
`Number.toInt32()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toInt64

Casts the input value to a signed 64-bit integer.

Usage	Returns
`Number.toInt64()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toInt8

Casts the input value to a signed 8-bit integer.

Usage	Returns
`Number.toInt8()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toLong

Casts the input value to a signed 64-bit integer.

Usage	Returns
`Number.toLong()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toShort

Casts the input value to a signed 16-bit integer.

Usage	Returns
`Number.toShort()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toUint16

Casts the input value to an unsigned 16-bit integer.

Usage	Returns
`Number.toUint16()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toUint32

Casts the input value to an unsigned 32-bit integer.

Usage	Returns
`Number.toUint32()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.toUint8

Casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Number.toUint8()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.trigamma

Computes the trigamma function of the input.

Usage	Returns
`Number.trigamma()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.uint16

Casts the input value to an unsigned 16-bit integer.

Usage	Returns
`Number.uint16()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.uint32

Casts the input value to an unsigned 32-bit integer.

Usage	Returns
`Number.uint32()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.uint8

Casts the input value to an unsigned 8-bit integer.

Usage	Returns
`Number.uint8()`	Number

Argument	Type	Details
this: `input`	Number	The input value.

ee.Number.unitScale

Scales the input so that the range of input values [min, max] becomes [0, 1]. Values outside the range are NOT clamped. If min == max, 0 is returned.

Usage	Returns
`Number.unitScale(min, max)`	Number

Argument	Type	Details
this: `number`	Number	
`min`	Float	
`max`	Float	

ee.PixelType

Returns a PixelType of the given precision with the given limits per element, and an optional dimensionality.

Usage	Returns
`ee.PixelType(precision, minValue, maxValue, dimensions)`	PixelType

Argument	Type	Details
`precision`	Object	The pixel precision, one of 'int', 'float', or 'double'.
`minValue`	*Number, default: null*	*The minimum value of pixels of this type. If precision is 'float' or 'double', this can be null, signifying negative infinity.*
`maxValue`	*Number, default: null*	*The maximum value of pixels of this type. If precision is 'float' or 'double', this can be null, signifying positive infinity.*
`dimensions`	*Integer, default: 0*	*The number of dimensions in which pixels of this type can vary; 0 is a scalar, 1 is a vector, 2 is a matrix, etc.*

ee.PixelType.dimensions

Returns the number of dimensions for this type. Will be 0 for scalar values and >= 1 for array values.

Usage	Returns
`PixelType.dimensions()`	Integer

Argument	Type	Details
this: `pixelType`	PixelType	

ee.PixelType.double

Returns the 64-bit floating point pixel type.

Usage	Returns
`ee.PixelType.double()`	PixelType

No arguments.

ee.PixelType.float

Returns the 32-bit floating point pixel type.

Usage	Returns
`ee.PixelType.float()`	PixelType

No arguments.

ee.PixelType.int16

Returns the 16-bit signed integer pixel type.

Usage	Returns
`ee.PixelType.int16()`	PixelType

No arguments.

ee.PixelType.int32

Returns the 32-bit signed integer pixel type.

Usage	Returns
`ee.PixelType.int32()`	PixelType

No arguments.

ee.PixelType.int64

Returns the 64-bit signed integer pixel type.

Usage	Returns
`ee.PixelType.int64()`	PixelType

No arguments.

ee.PixelType.int8

Returns the 8-bit signed integer pixel type.

Usage	Returns
`ee.PixelType.int8()`	PixelType

No arguments.

ee.PixelType.maxValue

Returns the maximum value of the PixelType.

Usage	Returns
`PixelType.maxValue()`	Number

Argument	Type	Details
this: `pixelType`	PixelType	

ee.PixelType.minValue

Returns the minimum value of the PixelType.

Usage	Returns
`PixelType.minValue()`	Number

Argument	Type	Details
this: `pixelType`	PixelType	

ee.PixelType.precision

Returns the precision of the PixelType. One of 'int', 'float', or 'double'.

Usage	Returns
`PixelType.precision()`	String

Argument	Type	Details
this: `pixelType`	PixelType	

ee.PixelType.uint16

Returns the 16-bit unsigned integer pixel type.

Usage	Returns
`ee.PixelType.uint16()`	PixelType

No arguments.

ee.PixelType.uint32

Returns the 32-bit unsigned integer pixel type.

Usage	Returns
`ee.PixelType.uint32()`	PixelType

No arguments.

ee.PixelType.uint8

Returns the 8-bit unsigned integer pixel type.

Usage	Returns
`ee.PixelType.uint8()`	PixelType

No arguments.

ee.Projection

Returns a Projection with the given base coordinate system and the given transform between projected coordinates and the base. If no transform is specified, the identity transform is assumed.

Usage	Returns
`ee.Projection(crs, transform, transformWkt)`	Projection

Argument	Type	Details
`crs`	Object	The base coordinate reference system of this Projection, given as a well-known authority code (e.g. 'EPSG:4326') or a WKT string.
`transform`	*List, default: null*	*The transform between projected coordinates and the base coordinate system, specified as a 2x3 affine transform matrix in row-major order: [xScale, xShearing, xTranslation, yShearing, yScale, yTranslation]. May not specify both this and 'transformWkt'.*
`transformWkt`	*String, default: null*	*The transform between projected coordinates and the base coordinate system, specified as a WKT string. May not specify both this and 'transform'.*

ee.Projection.atScale

Returns the projection scaled such that its units have the given scale in linear meters, as measured at the point of true scale.

Usage	Returns
`Projection.atScale(meters)`	Projection

Argument	Type	Details
this: `projection`	Projection	
`meters`	Float	

ee.Projection.crs

Returns the authority code (e.g. 'EPSG:4326') for the base coordinate system of this projection, or null if the base coordinate system is not found in any available database.

Usage	Returns
`Projection.crs()`	String

Argument	Type	Details
this: `projection`	Projection	

ee.Projection.nominalScale

Returns the linear scale in meters of the units of this projection, as measured at the point of true scale.

Usage	Returns
`Projection.nominalScale()`	Float

Argument	Type	Details
this: `proj`	Projection	

ee.Projection.scale

Returns the projection scaled by the given amount in each axis.

Usage	Returns
`Projection.scale(x, y)`	Projection

Argument	Type	Details
this: `projection`	Projection	
x	Float	
y	Float	

ee.Projection.transform

Returns a WKT representation of the transform of this Projection. This is the transform that converts from projected coordinates to the base coordinate system.

Usage	Returns
`Projection.transform()`	String

Argument	Type	Details
this: `projection`	Projection	

ee.Projection.translate

Returns the projection translated by the given amount in each axis.

Usage	Returns
`Projection.translate(x, y)`	Projection

Argument	Type	Details
this: `projection`	Projection	
x	Float	
y	Float	

ee.Projection.wkt

Returns a WKT representation of the base coordinate system of this Projection.

Usage	Returns
`Projection.wkt()`	String

Argument	Type	Details
this: `projection`	Projection	

ee.Reducer.allNonZero

Returns a Reducer that returns 1 if all of its inputs are non-zero, 0 otherwise.

Usage	Returns
`ee.Reducer.allNonZero()`	Reducer

No arguments.

ee.Reducer.anyNonZero

Returns a Reducer that returns 1 if any of its inputs are non-zero, 0 otherwise.

Usage	Returns
`ee.Reducer.anyNonZero()`	Reducer

No arguments.

ee.Reducer.autoHistogram

Create a reducer that will compute a histogram of the inputs. The output is a Nx2 array of the lower bucket bounds and the counts (or cumulative counts) of each bucket, and is suitable for use per-pixel.

Usage	Returns
`ee.Reducer.autoHistogram(`*`maxBuckets`*`, `*`minBucketWidth`*`, `*`maxRaw`*`, `*`cumulative`*`)`	Reducer

Argument	Type	Details
`maxBuckets`	*Integer, default: null*	*The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2.*
`minBucketWidth`	*Float, default: null*	*The minimum histogram bucket width, or null to allow any power of 2.*
`maxRaw`	*Integer, default: null*	*The number of values to accumulate before building the initial histogram.*
`cumulative`	*Boolean, default: false*	

ee.Reducer.bitwiseAnd

Returns a Reducer that computes the bitwise-and summation of its inputs.

Usage	Returns
`ee.Reducer.bitwiseAnd()`	Reducer

No arguments.

ee.Reducer.bitwiseOr

Returns a Reducer that computes the bitwise-or summation of its inputs.

Usage	Returns
`ee.Reducer.bitwiseOr()`	Reducer

No arguments.

ee.Reducer.centeredCovariance

Creates a reducer that reduces some number of 1-D arrays of the same length N to a covariance matrix of shape NxN. WARNING: this reducer requires that the data has been mean centered.

Usage	Returns
`ee.Reducer.centeredCovariance()`	Reducer

No arguments.

ee.Reducer.circularMean

Returns a Reducer that computes the (weighted) circular mean of its inputs, which are expected to be in radians. Output will be in the range (-π to π).

Usage	Returns
`ee.Reducer.circularMean()`	Reducer

No arguments.

ee.Reducer.circularStddev

Returns a Reducer that computes the (weighted) circular standard deviation of its inputs, which are expected to be in radians, using the sqrt(-2 * ln(R)) formula.

Usage	Returns
`ee.Reducer.circularStddev()`	Reducer

No arguments.

ee.Reducer.circularVariance

Returns a Reducer that computes the (weighted) circular variance of its inputs, which are expected to be in radians.

Usage	Returns
`ee.Reducer.circularVariance()`	Reducer

No arguments.

ee.Reducer.combine

Creates a Reducer that runs two reducers in parallel. The combined reducer's outputs will be those of reducer1 followed by those of reducer2, where the output names of reducer2 are prefixed with the given string. If sharedInputs is true, the reducers must have the same number of inputs, and the combined reducer's will match them; if it is false, the inputs of the combined reducer will be those of reducer1 followed by those of reducer2.

Usage	Returns
`Reducer.combine(reducer2, outputPrefix, sharedInputs)`	Reducer

Argument	Type	Details
this: `reducer1`	Reducer	
`reducer2`	Reducer	
`outputPrefix`	*String, default: ""*	*Prefix for reducer2's output names.*
`sharedInputs`	*Boolean, default: false*	

ee.Reducer.count

Returns a Reducer that computes the number of non-null inputs.

Usage	Returns
`ee.Reducer.count()`	Reducer

No arguments.

ee.Reducer.countDistinct

Returns a Reducer that computes the number of distinct inputs.

Usage	Returns
`ee.Reducer.countDistinct()`	Reducer

No arguments.

ee.Reducer.countDistinctNonNull

Returns a Reducer that computes the number of distinct inputs, ignoring nulls.

Usage	Returns
`ee.Reducer.countDistinctNonNull()`	Reducer

No arguments.

ee.Reducer.countEvery

Returns a Reducer that computes the number of inputs.

Usage	Returns
`ee.Reducer.countEvery()`	Reducer

No arguments.

ee.Reducer.countRuns

Returns a Reducer that computes the number of runs of distinct, non-null inputs.

Usage	Returns
`ee.Reducer.countRuns()`	Reducer

No arguments.

ee.Reducer.covariance

Creates a reducer that reduces some number of 1-D arrays of the same length N to a covariance matrix of shape NxN. This reducer uses the one-pass covariance formula from Sandia National Laboratories Technical Report SAND2008-6212, which can lose accuracy if the values span a large range.

Usage	Returns
`ee.Reducer.covariance()`	Reducer

No arguments.

ee.Reducer.disaggregate

Separates aggregate inputs (Arrays, Lists or Dictionaries) into individual items that are then each passed to the specified reducer. When used on dictionaries, the dictionary keys are ignored. Non-aggregated inputs (ie: numbers or strings) are passed to the underlying reducer directly.

Usage	Returns
`Reducer.disaggregate(`*`axis`*`)`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	The reducer for which to disaggregate inputs.
`axis`	*Integer, default: null*	*If specified, indicates an array axis along which to disaggregate. If not specified, arrays are completely disaggregated. Ignored for non-array types.*

ee.Reducer.first

Returns a Reducer that returns the first of its inputs.

Usage	Returns
`ee.Reducer.first()`	Reducer

No arguments.

ee.Reducer.firstNonNull

Returns a Reducer that returns the first of its non-null inputs.

Usage	Returns
`ee.Reducer.firstNonNull()`	Reducer

No arguments.

ee.Reducer.fixed2DHistogram

Creates a reducer that will compute a 2D histogram of the inputs using a fixed number of fixed width bins. Values outside of the [min, max) range on either axis are ignored. The output is a 2D array of counts, and 2 1-D arrays of bucket lower edges for the xAxis and the yXais. This reducer is suitable for use per-pixel, however it is always unweighted. The maximum count for any bucket is 2^31 - 1.

Usage	Returns
`ee.Reducer.fixed2DHistogram(xMin, xMax, xSteps, yMin, yMax, ySteps)`	Reducer

Argument	Type	Details
`xMin`	Float	The lower (inclusive) bound of the first bucket on the X axis.
`xMax`	Float	The upper (exclusive) bound of the last bucket on the X axis.
`xSteps`	Integer	The number of buckets to use on the X axis.
`yMin`	Float	The lower (inclusive) bound of the first bucket on the Y axis.
`yMax`	Float	The upper (exclusive) bound of the last bucket on the Y axis.
`ySteps`	Integer	The number of buckets to use on the Y axis.

ee.Reducer.fixedHistogram

Creates a reducer that will compute a histogram of the inputs using a fixed number of fixed width bins. Values outside of the [min, max) range are ignored. The output is a Nx2 array of bucket lower edges and counts (or cumulative counts) and is suitable for use per-pixel.

Usage	Returns
`ee.Reducer.fixedHistogram(min, max, steps, cumulative)`	Reducer

Argument	Type	Details
`min`	Float	The lower (inclusive) bound of the first bucket.
`max`	Float	The upper (exclusive) bound of the last bucket.
`steps`	Integer	The number of buckets to use.
`cumulative`	*Boolean, default: false*	*When true, generates a cumulative histogram.*

ee.Reducer.forEach

Creates a Reducer by combining a copy of the given reducer for each output name in the given list. If the reducer has a single output, the output names are used as-is; otherwise they are prefixed to the original output names.

Usage	Returns
`Reducer.forEach(outputNames)`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	
`outputNames`	List	

ee.Reducer.forEachBand

Creates a Reducer by combining a copy of the given reducer for each band in the given image, using the band names as output names.

Usage	Returns
`Reducer.forEachBand(image)`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	
`image`	Image	

ee.Reducer.forEachElement

Separately reduces each position in array inputs of equal shape, producing an array output of the same shape.

For example, with the 'sum' reducer applied to 5 arrays with shape 2x2, the output will be a 2x2 array, where each position is the sum of the 5 values at that position.

Usage	Returns
`Reducer.forEachElement()`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	The reducer to apply to each array element.

ee.Reducer.frequencyHistogram

Returns a Reducer that returns a (weighted) frequency table of its inputs.

Usage	Returns
`ee.Reducer.frequencyHistogram()`	Reducer

No arguments.

ee.Reducer.geometricMedian

Creates a reducer that computes the geometric median across a set of inputs.

Usage	Returns
`ee.Reducer.geometricMedian(numX, eta, initialStepSize)`	Reducer

Argument	Type	Details
`numX`	Integer	The number of input dimensions.
`eta`	*Float, default: 0.001*	*The minimum improvement in the solution used as a stopping criteria for the solver.*
`initialStepSize`	*Float, default: 10*	*The initial step size used in the solver.*

ee.Reducer.getOutputs

Returns a list of the output names of the given reducer.

Usage	Returns
`Reducer.getOutputs()`	List

Argument	Type	Details
this: `reducer`	Reducer	

ee.Reducer.group

Groups reducer records by the value of a given input, and reduces each group with the given reducer.

Usage	Returns
`Reducer.group(`*`groupField`*`, `*`groupName`*`)`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	The reducer to apply to each group, without the group field.
`groupField`	*Integer, default: 0*	*The field that contains record groups.*
`groupName`	*String, default: "group"*	*The dictionary key that contains the group. Defaults to 'group'.*

ee.Reducer.histogram

Create a reducer that will compute a histogram of the inputs.

Usage	Returns
`ee.Reducer.histogram(`*`maxBuckets`*`, `*`minBucketWidth`*`, `*`maxRaw`*`)`	Reducer

Argument	Type	Details
`maxBuckets`	*Integer, default: null*	*The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2.*
`minBucketWidth`	*Float, default: null*	*The minimum histogram bucket width, or null to allow any power of 2.*
`maxRaw`	*Integer, default: null*	*The number of values to accumulate before building the initial histogram.*

ee.Reducer.intervalMean

Creates a Reducer to compute the mean of all inputs in the specified percentile range. For small numbers of inputs (up to maxRaw) the mean will be computed directly; for larger numbers of inputs the mean will be derived from a histogram.

Usage	Returns
`ee.Reducer.intervalMean(minPercentile, maxPercentile, maxBuckets, minBucketWidth, maxRaw)`	Reducer

Argument	Type	Details
`minPercentile`	Float	The lower bound of the percentile range.
`maxPercentile`	Float	The upper bound of the percentile range.
`maxBuckets`	*Integer, default: null*	*The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2.*
`minBucketWidth`	*Float, default: null*	*The minimum histogram bucket width, or null to allow any power of 2.*
`maxRaw`	*Integer, default: null*	*The number of values to accumulate before building the initial histogram.*

ee.Reducer.kendallsCorrelation

Creates a reducer that computes the Kendall's Tau-b rank correlation. A positive tau value indicates an increasing trend; negative value indicates a decreasing trend. See https://commons.apache.org/proper/commons-math/javadocs/api-3.6/org/apache/commons/math3/stat/correlation/KendallsCorrelation.html for details.

Usage	Returns
`ee.Reducer.kendallsCorrelation(numInputs)`	Reducer

Argument	Type	Details
`numInputs`	*Integer, default: 1*	*The number of inputs to expect (1 or 2). If 1 is specified, automatically generates sequence numbers for the x value (meaning there can be no ties).*

ee.Reducer.kurtosis

Returns a Reducer that Computes the kurtosis of its inputs.

Usage	Returns
`ee.Reducer.kurtosis()`	Reducer

No arguments.

ee.Reducer.last

Returns a Reducer that returns the last of its inputs.

Usage	Returns
`ee.Reducer.last()`	Reducer

No arguments.

ee.Reducer.lastNonNull

Returns a Reducer that returns the last of its non-null inputs.

Usage	Returns
`ee.Reducer.lastNonNull()`	Reducer

No arguments.

ee.Reducer.linearFit

Returns a Reducer that computes the slope and offset for a (weighted) linear regression of 2 inputs. The inputs are expected to be x data followed by y data..

Usage	Returns
`ee.Reducer.linearFit()`	Reducer

No arguments.

ee.Reducer.linearRegression

Creates a reducer that computes a linear least squares regression with numX independent variables and numY dependent variables. Each input tuple will have values for the independent variables followed by the dependent variables. The first output is a coefficients array with dimensions (numX, numY); each column contains the coefficients for the corresponding dependent variable. The second output is a vector of the root mean square of the residuals of each dependent variable. Both outputs are null if the system is underdetermined, e.g. the number of inputs is less than or equal to numX.

Usage	Returns
`ee.Reducer.linearRegression(numX, numY)`	Reducer

Argument	Type	Details
`numX`	Integer	The number of input dimensions.
`numY`	*Integer, default: 1*	*The number of output dimensions.*

ee.Reducer.max

Creates a reducer that outputs the maximum value of its (first) input. If numInputs is greater than one, also outputs the corresponding values of the additional inputs.

Usage	Returns
`ee.Reducer.max(numInputs)`	Reducer

Argument	Type	Details
`numInputs`	*Integer, default: 1*	*The number of inputs.*

ee.Reducer.mean

Returns a Reducer that computes the (weighted) arithmetic mean of its inputs.

Usage	Returns
`ee.Reducer.mean()`	Reducer

No arguments.

ee.Reducer.median

Create a reducer that will compute the median of the inputs. For small numbers of inputs (up to maxRaw) the median will be computed directly; for larger numbers of inputs the median will be derived from a histogram.

Usage	Returns
`ee.Reducer.median(`*`maxBuckets`*`, `*`minBucketWidth`*`, `*`maxRaw`*`)`	Reducer

Argument	Type	Details
`maxBuckets`	*Integer, default: null*	*The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2.*
`minBucketWidth`	*Float, default: null*	*The minimum histogram bucket width, or null to allow any power of 2.*
`maxRaw`	*Integer, default: null*	*The number of values to accumulate before building the initial histogram.*

ee.Reducer.min

Creates a reducer that outputs the minimum value of its (first) input. If numInputs is greater than one, also outputs the corresponding values of the additional inputs.

Usage	Returns
`ee.Reducer.min(`*`numInputs`*`)`	Reducer

Argument	Type	Details
`numInputs`	*Integer, default: 1*	*The number of inputs.*

ee.Reducer.minMax

Returns a Reducer that computes the minimum and maximum of its inputs.

Usage	Returns
`ee.Reducer.minMax()`	Reducer

No arguments.

ee.Reducer.mode

Create a reducer that will compute the mode of the inputs. For small numbers of inputs (up to maxRaw) the mode will be computed directly; for larger numbers of inputs the mode will be derived from a histogram.

Usage	Returns
`ee.Reducer.mode(`*`maxBuckets`*`, `*`minBucketWidth`*`, `*`maxRaw`*`)`	Reducer

Argument	Type	Details
`maxBuckets`	*Integer, default: null*	*The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2.*
`minBucketWidth`	*Float, default: null*	*The minimum histogram bucket width, or null to allow any power of 2.*
`maxRaw`	*Integer, default: null*	*The number of values to accumulate before building the initial histogram.*

ee.Reducer.pearsonsCorrelation

Creates a two-input reducer that computes Pearson's product-moment correlation coefficient and the 2-sided p-value test for correlation = 0.

Usage	Returns
`ee.Reducer.pearsonsCorrelation()`	Reducer

No arguments.

ee.Reducer.percentile

Create a reducer that will compute the specified percentiles, e.g. given [0, 50, 100] will produce outputs named 'p0', 'p50', and 'p100' with the min, median, and max respectively. For small numbers of inputs (up to maxRaw) the percentiles will be computed directly; for larger numbers of inputs the percentiles will be derived from a histogram.

Usage	Returns
`ee.Reducer.percentile(percentiles, `*`outputNames`*`, `*`maxBuckets`*`, `*`minBucketWidth`*`, `*`maxRaw`*`)`	Reducer

Argument	Type	Details
`percentiles`	List	A list of numbers between 0 and 100.
`outputNames`	*List, default: null*	*A list of names for the outputs, or null to get default names.*
`maxBuckets`	*Integer, default: null*	*The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2.*
`minBucketWidth`	*Float, default: null*	*The minimum histogram bucket width, or null to allow any power of 2.*
`maxRaw`	*Integer, default: null*	*The number of values to accumulate before building the initial histogram.*

ee.Reducer.product

Returns a Reducer that computes the product of its inputs.

Usage	Returns
`ee.Reducer.product()`	Reducer

No arguments.

ee.Reducer.repeat

Creates a Reducer by combining the specified number of copies of the given reducer. Output names are the same as the given reducer, but each is a list of the corresponding output from each of the reducers.

Usage	Returns
`Reducer.repeat(count)`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	
`count`	Integer	

ee.Reducer.ridgeRegression

Creates a reducer that computes a ridge regression with numX independent variables (not including constant) followed by numY dependent variables. Ridge regression is a form of Tikhonov regularization which shrinks the regression coefficients by imposing a penalty on their size. With this implementation of ridge regression there NO NEED to include a constant value for bias.

The first output is a coefficients array with dimensions (numX + 1, numY); each column contains the coefficients for the corresponding dependent variable plus the intercept for the dependent variable in the last column. Additional outputs are a vector of the root mean square of the residuals of each dependent variable and a vector of p-values for each dependent variable. Outputs are null if the system is underdetermined, e.g. the number of inputs is less than numX + 1.

Usage	Returns
`ee.Reducer.ridgeRegression(numX, numY, lambda)`	Reducer

Argument	Type	Details
`numX`	Integer	the number of independent variables being regressed.
`numY`	*Integer, default: 1*	*the number of dependent variables.*
`lambda`	*Float, default: 0.1*	*Regularization parameter.*

ee.Reducer.robustLinearRegression

Creates a reducer that computes a robust least squares regression with numX independent variables and numY dependent variables, using iteratively reweighted least squares with the Talwar cost function. A point is considered an outlier if the RMS of residuals is greater than beta.

Each input tuple will have values for the independent variables followed by the dependent variables.

The first output is a coefficients array with dimensions (numX, numY); each column contains the coefficients for the corresponding dependent variable. The second is a vector of the root mean square of the residuals of each dependent variable. Both outputs are null if the system is underdetermined, e.g. the number of inputs is less than numX.

Usage	Returns
`ee.Reducer.robustLinearRegression(numX, numY, beta)`	Reducer

Argument	Type	Details
`numX`	Integer	The number of input dimensions.

Argument	Type	Details
`numY`	*Integer, default: 1*	*The number of output dimensions.*
`beta`	*Float, default: null*	*Residual error outlier margin. If null, a default value will be computed.*

ee.Reducer.sampleStdDev

Returns a Reducer that computes the sample standard deviation of its inputs.

Usage	Returns
`ee.Reducer.sampleStdDev()`	Reducer

No arguments.

ee.Reducer.sampleVariance

Returns a Reducer that computes the sample variance of its inputs.

Usage	Returns
`ee.Reducer.sampleVariance()`	Reducer

No arguments.

ee.Reducer.sensSlope

Creates a two-input reducer that computes the Sen's slope estimator. The inputs are expected to be x data followed by y data. It returns two double values; the estimated slope and the offset.

Usage	Returns
`ee.Reducer.sensSlope()`	Reducer

No arguments.

ee.Reducer.setOutputs

Returns a Reducer with the same inputs as the given Reducer, but with outputs renamed and/or removed.

Usage	Returns
`Reducer.setOutputs(outputs)`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	
`outputs`	List	The new output names; any output whose name is null or empty will be dropped.

ee.Reducer.skew

Returns a Reducer that Computes the skewness of its inputs.

Usage	Returns
`ee.Reducer.skew()`	Reducer

No arguments.

ee.Reducer.spearmansCorrelation

Creates a two-input reducer that computes the Spearman's rank-moment correlation. See https://commons.apache.org/proper/commons-math/javadocs/api-3.6/org/apache/commons/math3/stat/correlation/SpearmansCorrelation.html for details.

Usage	Returns
`ee.Reducer.spearmansCorrelation()`	Reducer

No arguments.

ee.Reducer.splitWeights

Returns a Reducer with the same outputs as the given Reducer, but with each weighted input replaced by two unweighted inputs.

Usage	Returns
`Reducer.splitWeights()`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	

ee.Reducer.stdDev

Returns a Reducer that computes the standard deviation of its inputs.

Usage	Returns
`ee.Reducer.stdDev()`	Reducer

No arguments.

ee.Reducer.sum

Returns a Reducer that computes the (weighted) sum of its inputs.

Usage	Returns
`ee.Reducer.sum()`	Reducer

No arguments.

ee.Reducer.toCollection

Returns a reducer that collects its inputs into a FeatureCollection.

Usage	Returns
`ee.Reducer.toCollection(propertyNames, numOptional)`	Reducer

Argument	Type	Details
`propertyNames`	List	The property names that will be defined on each output feature; determines the number of reducer inputs.
`numOptional`	*Integer, default: 0*	*The last numOptional inputs will be considered optional; the other inputs must be non-null or the input tuple will be dropped.*

ee.Reducer.toList

Creates a reducer that collects its inputs into a list, optionally grouped into tuples.

Usage	Returns
`ee.Reducer.toList(`*`tupleSize`*`, `*`numOptional`*`)`	Reducer

Argument	Type	Details
`tupleSize`	*Integer, default: null*	*The size of each output tuple, or null for no grouping. Also determines the number of inputs (null tupleSize has 1 input).*
`numOptional`	*Integer, default: 0*	*The last numOptional inputs will be considered optional; the other inputs must be non-null or the input tuple will be dropped.*

ee.Reducer.unweighted

Returns a Reducer with the same inputs and outputs as the given Reducer, but with no weighted inputs.

Usage	Returns
`Reducer.unweighted()`	Reducer

Argument	Type	Details
this: `reducer`	Reducer	

ee.Reducer.variance

Returns a Reducer that computes the variance of its inputs.

Usage	Returns
`ee.Reducer.variance()`	Reducer

No arguments.

ee.String

Constructs a new String.

Usage	Returns
`ee.String(string)`	String

Argument	Type	Details
`string`	Object|String	A string or a computed object.

ee.String.aside

Calls a function passing this object as the first argument, and returning itself. Convenient e.g. when debugging:

var c = ee.ImageCollection('foo').aside(print)

.filterDate('2001-01-01', '2002-01-01').aside(print, 'In 2001')

.filterBounds(geom).aside(print, 'In region')

.aside(Map.addLayer, {min: 0, max: 142}, 'Filtered')

.select('a', 'b');

Returns the same object, for chaining.

Usage	Returns
`String.aside(func, var_args)`	ComputedObject

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`func`	Function	The function to call.

Argument	Type	Details
`var_args`	VarArgs	Any extra arguments to pass to the function.

ee.String.cat

Concatenates two strings.

Usage	Returns
`String.cat(string2)`	String

Argument	Type	Details
this: `string1`	String	The first string.
`string2`	String	The second string.

ee.String.compareTo

Compares two strings lexicographically. Returns: the value 0 if the two strings are lexicographically equal; a value less than 0 if string1 is less than string2; and a value greater than 0 if string1 is lexicographically greater than string2.

Usage	Returns
`String.compareTo(string2)`	Integer

Argument	Type	Details
this: `string1`	String	The string to compare.
`string2`	String	The string to be compared.

ee.String.decodeJSON

Decodes a JSON string.

Usage	Returns
`String.decodeJSON()`	Object

Argument	Type	Details
this: `string`	String	The string to decode.

ee.String.encodeJSON

Encodes an object to JSON. Supports primitives, lists and dictionaries.

Usage	Returns
`ee.String.encodeJSON(object)`	String

Argument	Type	Details
`object`	Object	The object to encode.

ee.String.equals

Checks for string equality with a given object. Returns true if the target is a string and is lexicographically equal to the reference, or false otherwise.

Usage	Returns
`String.equals(target)`	Boolean

Argument	Type	Details
this: `reference`	String	The string to compare for equality.
`target`	Object	The second object to check for equality.

ee.String.evaluate

Asynchronously retrieves the value of this object from the server and passes it to the provided callback function.

Usage	Returns
`String.evaluate(callback)`	

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	Function	A function of the form function(success, failure), called when the server returns an answer. If the request succeeded, the success argument contains the evaluated result. If the request failed, the failure argument will contains an error message.

ee.String.getInfo

Retrieves the value of this object from the server.

If no callback function is provided, the request is made synchronously. If a callback is provided, the request is made asynchronously.

The asynchronous mode is preferred because the synchronous mode stops all other code (for example, the EE Code Editor UI) while waiting for the server. To make an asynchronous request, evaluate() is preferred over getInfo().

Returns the computed value of this object.

Usage	Returns
`String.getInfo(`*`callback`*`)`	Object

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.String.index

Searches a string for the first occurrence of a substring. Returns the index of the first match, or -1.

Usage	Returns
`String.index(pattern)`	Integer

Argument	Type	Details
this: `target`	String	The string to search.
`pattern`	String	The string to find.

ee.String.length

Returns the length of a string.

Usage	Returns
`String.length()`	Integer

Argument	Type	Details
this: `string`	String	The string from which to get the length.

ee.String.match

Matches a string against a regular expression. Returns a list of matching strings.

Usage	Returns
`String.match(regex, flags)`	List

Argument	Type	Details
this: `input`	String	The string in which to search.
`regex`	String	The regular expression to match.
`flags`	*String, default: ""*	*A string specifying a combination of regular expression flags, specifically one or more of: 'g' (global match) or 'i' (ignore case)*

ee.String.replace

Returns a new string with some or all matches of a pattern replaced.

Usage	Returns
`String.replace(regex, replacement, flags)`	String

Argument	Type	Details
this: `input`	String	The string in which to search.
`regex`	String	The regular expression to match.
`replacement`	String	The string that replaces the matched substring.
`flags`	*String, default: ""*	*A string specifying a combination of regular expression flags, specifically one or more of: 'g' (global match) or 'i' (ignore case)*

ee.String.rindex

Searches a string for the last occurrence of a substring. Returns the index of the first match, or -1.

Usage	Returns
`String.rindex(pattern)`	Integer

Argument	Type	Details
this: `target`	String	The string to search.
`pattern`	String	The string to find.

ee.String.serialize

Returns the serialized representation of this object.

Usage	Returns
`String.serialize(legacy)`	String

Argument	Type	Details
this: `computedobject`	ComputedObject	The ComputedObject instance.
`legacy`	*Boolean, optional*	*Enables legacy format.*

ee.String.slice

Returns a substring of the given string. If the specified range exceeds the length of the string, returns a shorter substring.

Usage	Returns
`String.slice(start, end)`	String

Argument	Type	Details
this: `string`	String	The string to subset.
`start`	Integer	The beginning index, inclusive. Negative numbers count backwards from the end of the string.
`end`	*Integer, default: null*	*The ending index, exclusive. Defaults to the length of the string. Negative numbers count backwards from the end of the string.*

ee.String.split

Splits a string on a regular expression, Returning a list of strings.

Usage	Returns
`String.split(regex, flags)`	List

Argument	Type	Details
this: `string`	String	The string to split.
`regex`	String	A regular expression to split on. If regex is the empty string, then the input string is split into individual characters.
`flags`	*String, default: ""*	*A string specifying the regular expression flag: 'i' (ignore case)*

ee.String.toLowerCase

Converts all of the characters in a string to lower case.

Usage	Returns
`String.toLowerCase()`	String

Argument	Type	Details
this: `string`	String	The string to convert to lower case.

ee.String.toUpperCase

Converts all of the characters in a string to upper case.

Usage	Returns
`String.toUpperCase()`	String

Argument	Type	Details
this: `string`	String	The string to convert to upper case.

ee.String.trim

Returns a string whose value is the original string, with any leading and trailing whitespace removed.

Usage	Returns
`String.trim()`	String

Argument	Type	Details
this: `string`	String	The string to trim.

ee.Terrain.aspect

Calculates aspect in degrees from a terrain DEM.

The local gradient is computed using the 4-connected neighbors of each pixel, so missing values will occur around the edges of an image.

Usage	Returns
`ee.Terrain.aspect(input)`	Image

Argument	Type	Details
`input`	Image	An elevation image, in meters.

ee.Terrain.fillMinima

Fills local minima. Only works on INT types.

Usage	Returns
`ee.Terrain.fillMinima(image, `*`borderValue`*`, `*`neighborhood`*`)`	Image

Argument	Type	Details
`image`	Image	The image to fill.
`borderValue`	*Long, default: null*	*The border value.*
`neighborhood`	*Integer, default: 50*	*The size of the neighborhood to compute over.*

ee.Terrain.hillShadow

Creates a shadow band, with output 1 where pixels are illumunated and 0 where they are shadowed. Takes as input an elevation band, azimuth and zenith of the light source in degrees, a neighborhood size, and whether or not to apply hysteresis when a shadow appears. Currently, this algorithm only works for Mercator projections, in which light rays are parallel.

Usage	Returns
`ee.Terrain.hillShadow(image, azimuth, zenith, `*`neighborhoodSize`*`, `*`hysteresis`*`)`	Image

Argument	Type	Details
`image`	Image	The image to which to apply the shadow algorithm, in which each pixel should represent an elevation in meters.
`azimuth`	Float	Azimuth in degrees.

Argument	Type	Details
`zenith`	Float	Zenith in degrees.
`neighborhoodSize`	*Integer, default: 0*	*Neighborhood size.*
`hysteresis`	*Boolean, default: false*	*Use hysteresis. Less physically accurate, but may generate better images.*

ee.Terrain.hillshade

Computes a simple hillshade from a DEM.

Usage	Returns
`ee.Terrain.hillshade(input, azimuth, elevation)`	Image

Argument	Type	Details
`input`	Image	An elevation image, in meters.
`azimuth`	*Float, default: 270*	*The illumination azimuth in degrees from north.*
`elevation`	*Float, default: 45*	*The illumination elevation in degrees.*

ee.Terrain.products

Calculates slope, aspect, and a simple hillshade from a terrain DEM.

Expects an image containing either a single band of elevation, measured in meters, or if there's more than one band, one named 'elevation'. Adds output bands named 'slope' and 'aspect' measured in degrees plus an unsigned byte output band named 'hillshade' for visualization. All other bands and metadata are copied from the input image. The local gradient is computed using the 4-connected neighbors of each pixel, so missing values will occur around the edges of an image.

Usage	Returns
`ee.Terrain.products(input)`	Image

Argument	Type	Details
`input`	Image	An elevation image, in meters.

ee.Terrain.slope

Calculates slope in degrees from a terrain DEM.

The local gradient is computed using the 4-connected neighbors of each pixel, so missing values will occur around the edges of an image.

Usage	Returns
`ee.Terrain.slope(input)`	Image

Argument	Type	Details
`input`	Image	An elevation image, in meters.

ee.apply

Call a function with a dictionary of named arguments.

Returns an object representing the called function. If the signature specifies a recognized return type, the returned value will be cast to that type.

Usage	Returns
`ee.apply(func, namedArgs)`	ComputedObject

Argument	Type	Details
`func`	Function|String	The function to call. Either an ee.Function object or the name of an API function.
`namedArgs`	Object	A dictionary of arguments to the function.

ee.call

Call a function with the given positional arguments.

Returns an object representing the called function. If the signature specifies a recognized return type, the returned value will be cast to that type.

Usage	Returns
`ee.call(func, var_args)`	ComputedObject

Argument	Type	Details
`func`	Function|String	The function to call. Either an ee.Function object or the name of an API function.
`var_args`	VarArgs	Positional arguments to pass to the function.

ee.data.authenticateViaOauth

Configures client-side authentication of EE API calls through the Google APIs Client Library for JavaScript. The library will be loaded automatically if it is not already loaded on the page. The user will be asked to grant the application identified by clientId access to their EE data if they have not done so previously.

This or another authentication method should be called before ee.initialize().

Note that if the user has not previously granted access to the application identified by the client ID, by default this will try to pop up a dialog window prompting the user to grant the required permission. However, this popup can be blocked by the browser. To avoid this, specify the opt_onImmediateFailed callback, and in it render an in-page login button, then call ee.data.authenticateViaPopup() from the click event handler of this button. This stops the browser from blocking the popup, as it is now the direct result of a user action.

The auth token will be refreshed automatically when possible. You can safely assume that all async calls will be sent with the appropriate credentials. For synchronous calls, however, you should check for an auth token with ee.data.getAuthToken() and call ee.data.refreshAuthToken() manually if there is none. The token refresh operation is asynchronous and cannot be performed behind-the-scenes on-demand prior to synchronous calls.

Usage	Returns
`ee.data.authenticateViaOauth(clientId, success, `*`error`*`, `*`extraScopes`*`, `*`onImmediateFailed`*`, `*`suppressDefaultScopes`*`)`	

Argument	Type	Details
`clientId`	String	The application's OAuth client ID, or null to disable authenticated calls. This can be obtained through the Google Developers Console. The project must have a JavaScript origin that corresponds to the domain where the script is running.
`success`	Function	The function to call if authentication succeeded.
`error`	*Function, optional*	*The function to call if authentication failed, passed the error message. If authentication in immediate (behind-the-scenes) mode fails and opt_onImmediateFailed is specified, that function is called instead of opt_error.*
`extraScopes`	*List, optional*	*Extra OAuth scopes to request.*

Argument	Type	Details
`onImmediateFailed`	*Function, optional*	*The function to call if automatic behind-the-scenes authentication fails. Defaults to ee.data.authenticateViaPopup(), bound to the passed callbacks.*
`suppressDefaultScopes`	*Boolean, optional*	*When true, only scopes specified in opt_extraScopes are requested; the default scopes are not requested unless explicitly specified in opt_extraScopes.*

ee.data.authenticateViaPopup

Shows a popup asking for the user's permission. Should only be called if ee.data.authenticate() called its opt_onImmediateFailed argument in the past.

May be blocked by pop-up blockers if called outside a user-initiated handler.

Usage	Returns
`ee.data.authenticateViaPopup(`*`success`*`, `*`error`*`)`	

Argument	Type	Details
`success`	*Function, optional*	*The function to call if authentication succeeds.*
`error`	*Function, optional*	*The function to call if authentication fails, passing the error message.*

ee.data.authenticateViaPrivateKey

Configures server-side authentication of EE API calls through the Google APIs Node.js Client. Private key authentication is strictly for server-side API calls: for browser-based applications, use ee.data.authenticateViaOauth(). No user interaction (e.g. authentication popup) is necessary when using server-side authentication.

This or another authentication method should be called before ee.initialize().

The auth token will be refreshed automatically when possible. You can safely assume that all async calls will be sent with the appropriate credentials. For synchronous calls, however, you should check for an auth token with ee.data.getAuthToken() and call ee.data.refreshAuthToken() manually if there is none. The token refresh operation is asynchronous and cannot be performed behind-the-scenes, on demand, prior to synchronous calls.

Usage	Returns
`ee.data.authenticateViaPrivateKey(privateKey, `*`success`*`, `*`error`*`, `*`extraScopes`*`, `*`suppressDefaultScopes`*`)`	

Argument	Type	Details
`privateKey`	AuthPrivateKey	JSON content of private key.
`success`	*Function, optional*	*The function to call if authentication succeeded.*
`error`	*Function, optional*	*The function to call if authentication failed, passed the error message.*
`extraScopes`	*List, optional*	*Extra OAuth scopes to request.*
`suppressDefaultScopes`	*Boolean, optional*	*When true, only scopes specified in opt_extraScopes are requested; the default scopes are not not requested unless explicitly specified in opt_extraScopes.*

ee.data.cancelOperation

Cancels the given operation(s).

Usage	Returns
`ee.data.cancelOperation(operationName, callback)`	

Argument	Type	Details
`operationName`	List	Operation name(s).
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. The callback is passed an empty object.*

ee.data.computeValue

Sends a request to compute a value.

Returns result

Usage	Returns
`ee.data.computeValue(obj, callback)`	Object|Value

Argument	Type	Details
`obj`	Object	

Argument	Type	Details
`callback`	*Function, optional*	

ee.data.copyAsset

Copies the asset from sourceId into destinationId.

Usage	Returns
`ee.data.copyAsset(sourceId, destinationId, overwrite, callback)`	

Argument	Type	Details
`sourceId`	String	The ID of the asset to copy.
`destinationId`	String	The ID of the new asset created by copying.
`overwrite`	*Boolean, optional*	*Overwrite any existing destination asset ID.*
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. The callback is passed an empty object and an error message, if any.*

ee.data.createAsset

Creates an asset from a JSON value. To create an empty image collection or folder, pass in a "value" object with a "type" key whose value is one of ee.data.AssetType.* (i.e. "ImageCollection" or "Folder").

Returns a description of the saved asset, including a generated ID, or null if a callback is specified.

Usage	Returns
`ee.data.createAsset(value, path, force, properties, callback)`	Object

Argument	Type	Details
`value`	Object	An object describing the asset to create.
`path`	*String, optional*	*An optional desired ID, including full path.*
`force`	*Boolean, optional*	*Force overwrite.*
`properties`	*Object, optional*	*The keys and values of the properties to set*

Argument	Type	Details
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.createAssetHome

Attempts to create a home root folder (e.g. "users/joe") for the current user. This results in an error if the user already has a home root folder or the requested ID is unavailable.

Usage	Returns
`ee.data.createAssetHome(requestedId, `*`callback`*`)`	

Argument	Type	Details
`requestedId`	String	The requested ID of the home folder (e.g. "users/joe").
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.createFolder

Creates an asset folder.

Returns a description of the newly created folder.

Usage	Returns
`ee.data.createFolder(path, `*`force`*`, `*`callback`*`)`	Object

Argument	Type	Details
`path`	String	The path of the folder to create.
`force`	*Boolean, optional*	*Force overwrite.*
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.deleteAsset

Deletes the asset with the given id.

Usage	Returns
`ee.data.deleteAsset(assetId, callback)`	

Argument	Type	Details
`assetId`	String	The ID of the asset to delete.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. The callback is passed an empty object and an error message, if any.*

ee.data.getAsset

Load info for an asset, given an asset id.

Returns the value call results, or null if a callback is specified.

Usage	Returns
`ee.data.getAsset(id, callback)`	Object

Argument	Type	Details
`id`	String	The asset to be retrieved.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getAssetAcl

Returns the access control list of the asset with the given ID.

The authenticated user must be a writer or owner of an asset to see its ACL.

Usage	Returns
`ee.data.getAssetAcl(assetId, callback)`	AssetAcl

Argument	Type	Details
`assetId`	String	The ID of the asset to check.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getAssetRootQuota

Returns quota usage details for the asset root with the given ID.

Usage notes:

- The id *must* be a root folder like "users/foo" (not "users/foo/bar").

- The authenticated user must own the asset root to see its quota usage.

Usage	Returns
`ee.data.getAssetRootQuota(rootId, callback)`	AssetQuotaDetails

Argument	Type	Details
`rootId`	String	The ID of the asset root to check, e.g. "users/foo".
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getDownloadId

Get a Download ID.

Returns a download id and token, or null if a callback is specified.

Usage	Returns
`ee.data.getDownloadId(params, callback)`	DownloadId

Argument	Type	Details
`params`	Object	An object containing download options with the following possible values: `name:` a base name to use when constructing filenames. Only applicable when format is "ZIPPED_GEO_TIFF" (default) or filePerBand is true. Defaults to the image id (or "download" for computed images) when format is "ZIPPED_GEO_TIFF" or filePerBand is true, otherwise a random character string is generated. Band names are appended when filePerBand is true. `bands:` a description of the bands to download. Must be an array of band names or an array of dictionaries, each with the following keys (optional parameters apply only when filePerBand is true): `id:` the name of the band, a string, required. `crs:` an optional CRS string defining the band projection. `crs_transform:` an optional array of 6 numbers specifying an affine transform from the specified CRS, in row-major order: [xScale, xShearing, xTranslation, yShearing, yScale, yTranslation]

Argument	Type	Details
		`dimensions:` an optional array of two integers defining the width and height to which the band is cropped. `scale:` an optional number, specifying the scale in meters of the band; ignored if crs and crs_transform are specified.
		`crs:` a default CRS string to use for any bands that do not explicitly specify one.
		`crs_transform:` a default affine transform to use for any bands that do not specify one, of the same format as the `crs_transform` of bands.
		`dimensions:` default image cropping dimensions to use for any bands that do not specify them.
		`scale:` a default scale to use for any bands that do not specify one; ignored if `crs` and `crs_transform` are specified.
		`region:` a polygon specifying a region to download; ignored if `crs` and `crs_transform` is specified.
		`filePerBand:` whether to produce a separate GeoTIFF per band (boolean). Defaults to true. If false, a single GeoTIFF is produced and all band-level transformations will be ignored.
		`format:` the download format. One of: • "ZIPPED_GEO_TIFF" (GeoTIFF file(s) wrapped in a zip file, default) • "GEO_TIFF" (GeoTIFF file) • "NPY" (NumPy binary format) If "GEO_TIFF" or "NPY", filePerBand and all band-level transformations will be ignored. Loading a NumPy output results in a structured array.
		`id:` deprecated, use image parameter.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getFeatureViewTilesKey

Get a tiles key for a given map or asset. The tiles key can be passed to an instance of FeatureViewTileSource which can be rendered on a base map outside of the Code Editor.

Returns the call results. Null if a callback is specified.

Usage	Returns
`ee.data.getFeatureViewTilesKey(params, `*`callback`*`)`	FeatureViewTilesKey

Argument	Type	Details
`params`	FeatureViewVisualizationParameters	The visualization parameters as a (client-side) JavaScript object. For FeatureView assets: `assetId` (string) The asset ID for which to obtain a tiles key. `visParams` (Object) The visualization parameters for this layer.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getFilmstripThumbId

Get a Filmstrip Thumbnail Id for a given asset.

Returns the thumb ID and optional token, or null if a callback is specified.

Usage	Returns
`ee.data.getFilmstripThumbId(params, callback)`	ThumbnailId

Argument	Type	Details
`params`	FilmstripThumbnailOptions	Parameters to make the request with.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getMapId

Get a Map ID for a given asset

Returns the mapId call results, which may be passed to ee.data.getTileUrl or ui.Map.addLayer. Null if a callback is specified.

Usage	Returns
`ee.data.getMapId(params, callback)`	RawMapId

Argument	Type	Details
`params`	ImageVisualizationParameters	The visualization parameters as a (client-side) JavaScript object. For Images and ImageCollections: `image` (JSON string) The image to render.

Argument Type	Details
	`version` (number) Version number of image (or latest).
	`bands` (comma-separated strings) Comma-delimited list of band names to be mapped to RGB.
	`min` (comma-separated numbers) Value (or one per band) to map onto 00.
	`max` (comma-separated numbers) Value (or one per band) to map onto FF.
	`gain` (comma-separated numbers) Gain (or one per band) to map onto 00-FF.
	`bias` (comma-separated numbers) Offset (or one per band) to map onto 00-FF.
	`gamma` (comma-separated numbers) Gamma correction factor (or one per band).
	`palette` (comma-separated strings) List of CSS-style color strings (single-band previews only).
	`opacity` (number) a number between 0 and 1 for opacity.
	`format` (string) Either "jpg" or "png".
`callback`*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getOperation

Gets information on an operation or list of operations.

See more details on Operations here:
https://cloud.google.com/apis/design/design_patterns#long_running_operations

Returns operation status, or a map from operation names to status. Each Operation contains:

- name: operation name in the format projects/X/operations/Y

- done: true when operation has finished running.

- error: may be set when done=true. Contains message and other fields from
https://cloud.google.com/tasks/docs/reference/rpc/google.rpc#status

- metadata, which contains

  + state: PENDING, RUNNING, CANCELLING, SUCCEEDED, CANCELLED, or FAILED

  + description: Supplied task description

  + type: EXPORT_IMAGE, EXPORT_FEATURES, etc.

  + create_time: Time the operation was first submitted.

+ update_time: Timestamp of most recent update.

+ start_time: Time the operation started, when so.

+ end_time: Time the operation finished running, when so.

+ attempt: Number of retries of this task, starting at 1.

+ destination_uris: Resources output by this operation.

+ batch_eecu_usage_seconds: CPU used by this operation.

Usage	Returns
`ee.data.getOperation(operationName, callback)`	Dictionary|api.Operation

Argument	Type	Details
`operationName`	List	Operation name(s).
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getTableDownloadId

Get a download ID.

Returns a download id and token, or null if a callback is specified.

Usage	Returns
`ee.data.getTableDownloadId(params, callback)`	DownloadId

Argument	Type	Details
`params`	Object	An object containing table download options with the following possible values: `table:` The feature collection to download. `format:` The download format, CSV, JSON, KML, KMZ or TF_RECORD. `selectors:` List of strings of selectors that can be used to determine which attributes will be downloaded. `filename:` The name of the file that will be downloaded.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getThumbId

Get a Thumbnail Id for a given asset.

Returns the thumb ID and optional token, or null if a callback is specified.

Usage	Returns
`ee.data.getThumbId(params, callback)`	ThumbnailId

Argument	Type	Details
`params`	ThumbnailOptions	An object containing thumbnail options with the following possible values: `image` (ee.Image) The image to make a thumbnail. `bands` (array of strings) An array of band names. `format` (string) The file format ("png", "jpg", "geotiff"). `name` (string): The base name. Use ee.Image.getThumbURL for region, dimensions, and visualization options support.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getTileUrl

Generate a URL for map tiles from a Map ID and coordinates. If formatTileUrl is not present, we generate it by using or guessing the urlFormat string, and add urlFormat and formatTileUrl to `id` for future use.

Returns the tile URL.

Usage	Returns
`ee.data.getTileUrl(id, x, y, z)`	String

Argument	Type	Details
`id`	RawMapId	The Map ID to generate tiles for.
`x`	Number	The tile x coordinate.
`y`	Number	The tile y coordinate.
`z`	Number	The tile zoom level.

ee.data.getVideoThumbId

Get a Video Thumbnail Id for a given asset.

Returns the thumb ID and optional token, or null if a callback is specified.

Usage	Returns
`ee.data.getVideoThumbId(params, callback)`	ThumbnailId

Argument	Type	Details
`params`	VideoThumbnailOptions	Parameters to make the request with.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.getWorkloadTag

Returns the currently set workload tag.

Usage	Returns
`ee.data.getWorkloadTag()`	String

No arguments.

ee.data.listAssets

Returns a list of the contents in an asset collection or folder, in an object that includes an `assets` array and an optional `nextPageToken`.

Usage	Returns
`ee.data.listAssets(parent, params, callback)`	api.ListAssetsResponse

Argument	Type	Details
`parent`	String	The ID of the collection or folder to list.
`params`	*api.ProjectsAssetsListAssetsNamedParameters, optional*	*An object containing optional request parameters with the following possible values:*

Argument	Type	Details
		`pageSize` (string) The number of results to return. Defaults to 1000.
		`pageToken` (string) The token for the page of results to return.
		`filter` (string) An additional filter query to apply. Example query: `properties.my_property>=1 AND properties.my_property<2 AND startTime >= "2019-01-01T00:00:00.000Z" AND endTime < "2020-01-01T00:00:00.000Z" AND intersects("{'type':'Point','coordinates':[0,0]}")` See https://google.aip.dev/160 for how to construct a query.
		`view` (string) Specifies how much detail is returned in the list. Either "FULL" (default) for all image properties or "BASIC".
`callback`	*Function, optional*	*If not supplied, the call is made synchronously.*

ee.data.listBuckets

Returns top-level assets and folders for the Cloud Project or user. Leave the project field blank to use the current project.

Usage	Returns
`ee.data.listBuckets(`*`project`*`, `*`callback`*`)`	api.ListAssetsResponse

Argument	Type	Details
`project`	*String, optional*	*Project to query, e.g. "projects/my-project". Defaults to current project. Use "projects/earthengine-legacy" for user home folders.*
`callback`	*Function, optional*	*If not supplied, the call is made synchronously.*

ee.data.listFeatures

List features for a given table asset.

Returns the call results. Null if a callback is specified.

Usage	Returns
`ee.data.listFeatures(asset, params, callback)`	api.ListFeaturesResponse

Argument	Type	Details
`asset`	String	The table asset ID to query.
`params`	api.ProjectsAssetsListFeaturesNamedParameters	An object containing request parameters with the following possible values: `pageSize` (number): An optional maximum number of results per page, default is 1000. `pageToken` (string): An optional token identifying a page of results the server should return, usually taken from the response object. `region` (string): If present, a geometry defining a query region, specified as a GeoJSON geometry string (see RFC 7946). `filter` (comma-separated strings): If present, specifies additional simple property filters (see https://google.aip.dev/160).
`callback`	*Function, optional*	*An optional callback, called with two parameters: the first is the resulting list of features and the second is an error string on failure. If not supplied, the call is made synchronously.*

ee.data.listImages

Returns a list of the contents in an image collection, in an object that includes an `images` array and an optional `nextPageToken`.

Usage	Returns
`ee.data.listImages(parent, params, callback)`	ListImagesResponse

Argument	Type	Details
`parent`	String	The ID of the image collection to list.
`params`	*Object, optional*	*An object containing optional request parameters with the following possible values:* `pageSize` (string) The number of results to return. Defaults to 1000. `pageToken` (string) The token page of results to return. `startTime` (ISO 8601 string) The minimum start time (inclusive).

Argument	Type	Details
		`endTime` (ISO 8601 string) The maximum end time (exclusive).
		`region` (GeoJSON or WKT string) A region to filter on.
		`properties` (list of strings) A list of property filters to apply, for example: ["classification=urban", "size>=2"].
		`filter` (string) An additional filter query to apply. Example query: `properties.my_property>=1 AND properties.my_property<2 AND startTime >= "2019-01-01T00:00:00.000Z" AND endTime < "2020-01-01T00:00:00.000Z" AND intersects("{'type':'Point','coordinates':[0,0]}")` See https://google.aip.dev/160 for how to construct a query.
		`view` (string) Specifies how much detail is returned in the list. Either "FULL" (default) for all image properties or "BASIC".
`callback`	*Function, optional*	*If not supplied, the call is made synchronously.*

ee.data.listOperations

Returns see getOperation for details on the Operation object.

Usage	Returns
`ee.data.listOperations(`*`limit`*`, `*`callback`*`)`	List

Argument	Type	Details
`limit`	*Number, optional*	*Maximum number of results to return.*
`callback`	*Function, optional*	

ee.data.makeDownloadUrl

Create a download URL from a docid and token.

Returns the download URL.

Usage	Returns
`ee.data.makeDownloadUrl(id)`	String

Argument	Type	Details
`id`	DownloadId	A download id and token.

ee.data.makeTableDownloadUrl

Create a table download URL from a docid and token.

Returns the download URL.

Usage	Returns
`ee.data.makeTableDownloadUrl(id)`	String

Argument	Type	Details
`id`	DownloadId	A table download id and token.

ee.data.makeThumbUrl

Create a thumbnail URL from a thumbid and token.

Returns the thumbnail URL.

Usage	Returns
`ee.data.makeThumbUrl(id)`	String

Argument	Type	Details
`id`	ThumbnailId	A thumbnail ID and token.

ee.data.newTaskId

Generates an "unsubmitted" ID for a long-running task.

Before tasks are submitted, they may be assigned IDs to track them. The server ensures that the same ID cannot be reused. These IDs have a UUID format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.

Tasks that are running on the server have a ID without hyphens. This is returned by ee.data.startProcessing and other batch methods.

Returns an array containing generated ID strings, or null if a callback is specified.

Usage	Returns
`ee.data.newTaskId(`*`count`*`, `*`callback`*`)`	List

Argument	Type	Details
`count`	*Number, optional*	*The number of IDs to generate, one by default.*
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.renameAsset

Renames the asset from sourceId to destinationId.

Usage	Returns
`ee.data.renameAsset(sourceId, destinationId, `*`callback`*`)`	

Argument	Type	Details
`sourceId`	String	The ID of the asset to rename.
`destinationId`	String	The new ID of the asset.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. The callback is passed an empty object and an error message, if any.*

ee.data.resetWorkloadTag

Resets the tag back to the default. If resetDefault parameter is set to true, the default will be set to empty before resetting.

Usage	Returns
`ee.data.resetWorkloadTag(`*`resetDefault`*`)`	

Argument	Type	Details
`resetDefault`	*Boolean, optional*	

ee.data.setAssetAcl

Sets the access control list of the asset with the given ID.

The owner ACL cannot be changed, and the final ACL of the asset is constructed by merging the OWNER entries of the old ACL with the incoming ACL record.

The authenticated user must be a writer or owner of an asset to set its ACL.

Usage	Returns
`ee.data.setAssetAcl(assetId, aclUpdate, `*`callback`*`)`	

Argument	Type	Details
`assetId`	String	The ID of the asset to set the ACL on.
`aclUpdate`	AssetAclUpdate	The updated ACL.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously. The callback is passed an empty object.*

ee.data.setDefaultWorkloadTag

Sets the workload tag, and as the default for which to reset back to.

For example, calling `ee.data.resetWorkloadTag()` will reset the workload tag back to the default chosen here. To reset the default back to none, pass in an empty string or pass in true to `ee.data.resetWorkloadTag(true)`, like so.

Workload tag must be 1 - 63 characters, beginning and ending with an alphanumeric character ([a-z0-9A-Z]) with dashes (-), underscores (_), dots

(.), and alphanumerics between, or an empty string to reset the default back to none.

Usage	Returns
`ee.data.setDefaultWorkloadTag(tag)`	

Argument	Type	Details
`tag`	String	

ee.data.setWorkloadTag

Sets the workload tag, used to label computation and exports.

Workload tag must be 1 - 63 characters, beginning and ending with an alphanumeric character ([a-z0-9A-Z]) with dashes (-), underscores (_), dots

(.), and alphanumerics between, or an empty string to clear the workload tag.

Usage	Returns
`ee.data.setWorkloadTag(tag)`	

Argument	Type	Details
`tag`	String	

ee.data.startIngestion

Creates an image asset ingestion task.

See ee.data.startProcessing for details on task IDs and response format.

Usage	Returns
`ee.data.startIngestion(taskId, request, `*`callback`*`)`	ProcessingResponse

Argument	Type	Details
`taskId`	String	Unsubmitted ID for the task (obtained from newTaskId).
`request`	IngestionRequest	The object that describes the ingestion.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.startProcessing

Create processing task that exports or pre-renders an image.

Return value is null if a callback is specified.

Returns an object with fields:

- taskId: Submitted task ID (without hyphens).

- name: Full operation name in the format projects/X/operations/Y

- started: will be 'OK'

- note: may have value 'ALREADY_EXISTS' if an identical task with the same unsubmitted ID already exists.

Usage	Returns
`ee.data.startProcessing(taskId, params, callback)`	ProcessingResponse

Argument	Type	Details
`taskId`	String	Unsubmitted ID for the task (obtained from newTaskId). Used to identify duplicated tasks; may be null. The server will create and return a submitted ID.
`params`	Object	The object that describes the processing task; only fields that are common for all processing types are documented here. type (string) Either 'EXPORT_IMAGE', 'EXPORT_FEATURES', 'EXPORT_VIDEO' or 'EXPORT_TILES'. json (string) JSON description of the image.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.startTableIngestion

Creates a table asset ingestion task.

See ee.data.startProcessing for details on task IDs and response format.

Usage	Returns
`ee.data.startTableIngestion(taskId, request, callback)`	ProcessingResponse

Argument	Type	Details
`taskId`	String	Unsubmitted ID for the task (obtained from newTaskId).
`request`	TableIngestionRequest	The object that describes the ingestion.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.updateAsset

Updates an asset.

The authenticated user must be a writer or owner of the asset.

Usage	Returns
`ee.data.updateAsset(assetId, asset, updateFields, callback)`	Object

Argument	Type	Details
`assetId`	String	The ID of the asset to update.
`asset`	api.EarthEngineAsset	The updated version of the asset, containing only the new values of the fields to be updated. Only the "start_time", "end_time", and "properties" fields can be updated. If a value is named in "updateMask", but is unset in "asset", then that value will be deleted from the asset.
`updateFields`	List	A list of the field names to update. This may contain: "start_time" or "end_time" to update the corresponding timestamp, "properties.PROPERTY_NAME" to update a given property, or "properties" to update all properties. If the list is empty, all properties and both timestamps will be updated.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.data.updateTask

Update one or more tasks' properties. For now, only the following properties may be updated: State (to CANCELLED)

Returns an array of updated tasks, or null if a callback is specified.

Usage	Returns
`ee.data.updateTask(taskId, action, callback)`	List

Argument	Type	Details
`taskId`	List	Submitted ID of the task or an array of multiple task IDs. May also contain operation names.
`action`	TaskUpdateActions	Action performed on tasks.
`callback`	*Function, optional*	*An optional callback. If not supplied, the call is made synchronously.*

ee.initialize

Initialize the library. If this hasn't been called by the time any object constructor is used, it will be called then. If this is called a second time with a different baseurl or tileurl, this doesn't do an un-initialization of e.g.: the previously loaded Algorithms, but will overwrite them and let point at alternate servers.

If initialize() is first called in asynchronous mode (by passing a success callback), any future asynchronous mode calls will add their callbacks to a queue and all the callbacks will be run together.

If a synchronous mode call is made after any number of asynchronous calls, it will block and execute all the previously supplied callbacks before returning.

In most cases, an authorization token should be set before the library is initialized, either with ee.data.authorize() or ee.data.setAuthToken().

In Python, this method is named ee.Initialize, with a capital I. Note that some parameters differ between JavaScript and Python. In addition to opt_url and project below, Python also supports: credentials - a google.oauth2.Credentials object or 'persistent' to use stored credentials (the default); http_transport - a httplib2.Http client.

Usage	Returns
`ee.initialize(`*`baseurl`*`, `*`tileurl`*`, `*`successCallback`*`, `*`errorCallback`*`, `*`xsrfToken`*`, `*`project`*`)`	

Argument	Type	Details
`baseurl`	*String, optional*	*The Earth Engine REST API endpoint. (Python argument name: opt_url)*
`tileurl`	*String, optional*	*The Earth Engine REST tile endpoint, this is optional and defaults to baseurl. (JavaScript only)*
`successCallback`	*Function, optional*	*An optional callback to be invoked when the initialization is successful. If not provided, the initialization is done synchronously. (JavaScript only)*
`errorCallback`	*Function, optional*	*An optional callback to be invoked with an error if the initialization fails. (JavaScript only)*
`xsrfToken`	*String, optional*	*A string to pass in the "xsrfToken" parameter of EE API XHRs. (JavaScript only)*
`project`	*String, optional*	*Optional client project ID or number to use when making API calls. (Python argument name: project)*

ee.reset

Reset the library to its base state. Useful for re-initializing to a different server.

Usage	Returns
`ee.reset()`	

No arguments.

Export.classifier.toAsset

Creates a batch task to export an ee.Classifier as an Earth Engine asset.

Usage	Returns
`Export.classifier.toAsset(classifier, description, assetId)`	

Argument	Type	Details
`classifier`	ComputedObject	The classifier to export.
`description`	*String, optional*	*A human-readable name of the task. Defaults to "myExportClassifierTask".*
`assetId`	*String, optional*	*The destination asset ID.*

Export.image.toAsset

Creates a batch task to export an Image as a raster to an Earth Engine asset. Tasks can be started from the Tasks tab.

Usage	Returns
`Export.image.toAsset(image, description, assetId, pyramidingPolicy, dimensions, region, scale, crs, crsTransform, maxPixels, shardSize, priority)`	

Argument	Type	Details
`image`	Image	The image to export.
`description`	*String, optional*	*A human-readable name of the task. Defaults to "myExportImageTask".*
`assetId`	*String, optional*	*The destination asset ID.*
`pyramidingPolicy`	*Object, optional*	*The pyramiding policy to apply to each band in the image, keyed by band name. Values must be one of: mean, sample, min, max, or mode. Defaults to "mean". A special*

Argument	Type	Details
		key, ".default" may be used to change the default for all bands.
`dimensions`	*Number|String, optional*	*The dimensions to use for the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.*
`region`	*Geometry.LinearRing|Geometry.Polygon|String, optional*	*A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string.*
`scale`	*Number, optional*	*Resolution in meters per pixel. Defaults to 1000.*
`crs`	*String, optional*	*CRS to use for the exported image.*
`crsTransform`	*List, optional*	*Affine transform to use for the exported image. Requires "crs" to be defined.*
`maxPixels`	*Number, optional*	*Restrict the number of pixels in the export. By default, you will see an error if the export exceeds 1e8 pixels. Setting this value explicitly allows one to raise or lower this limit.*
`shardSize`	*Number, optional*	*Size in pixels of the tiles in which this image will be computed. Defaults to 256.*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.image.toCloudStorage

Creates a batch task to export an Image as a raster to Google Cloud Storage. Tasks can be started from the Tasks tab.

"crsTransform", "scale", and "dimensions" are mutually exclusive.

Usage	Returns
`Export.image.toCloudStorage(image, `*`description`*`, `*`bucket`*`, `*`fileNamePrefix`*`, `*`dimensions`*`, `*`region`*`, `*`scale`*`, `*`crs`*`, `*`crsTransform`*`, `*`maxPixels`*`, `*`shardSize`*`, `*`fileDimensions`*`, `*`skipEmptyTiles`*`, `*`fileFormat`*`, `*`formatOptions`*`, `*`priority`*`)`	

Argument	Type	Details
`image`	Image	The image to export.

Argument	Type	Details
`description`	*String, optional*	*A human-readable name of the task. Defaults to "myExportImageTask".*
`bucket`	*String, optional*	*The Cloud Storage destination bucket.*
`fileNamePrefix`	*String, optional*	*The string used as the output's prefix. A trailing "/" indicates a path. Defaults to the task's description.*
`dimensions`	*Number|String, optional*	*The dimensions to use for the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.*
`region`	*Geometry.LinearRing|Geometry.Polygon|String, optional*	*A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string.*
`scale`	*Number, optional*	*Resolution in meters per pixel. Defaults to 1000.*
`crs`	*String, optional*	*CRS to use for the exported image.*
`crsTransform`	*List, optional*	*Affine transform to use for the exported image. Requires "crs" to be defined.*
`maxPixels`	*Number, optional*	*Restrict the number of pixels in the export. By default, you will see an error if the export exceeds 1e8 pixels. Setting this value explicitly allows one to raise or lower this limit.*
`shardSize`	*Number, optional*	*Size in pixels of the tiles in which this image will be computed. Defaults to 256.*
`fileDimensions`	*List, optional*	*The dimensions in pixels of each image file, if the image is too large to fit in a single file. May specify a single number to indicate a square shape, or an array of two dimensions to indicate (width,height). Note that the image will still be clipped to the overall image dimensions. Must be a multiple of shardSize.*
`skipEmptyTiles`	*Boolean, optional*	*If true, skip writing empty (i.e. fully-masked) image tiles. Defaults to false. Only supported on GeoTIFF exports.*
`fileFormat`	*String, optional*	*The string file format to which the image is exported. Currently only 'GeoTIFF' and 'TFRecord' are supported, defaults to 'GeoTIFF'.*
`formatOptions`	*ImageExportFormatConfig, optional*	*A dictionary of string keys to format-specific options. For 'GeoTIFF': 'cloudOptimized' (Boolean), 'noData' (float). For 'TFRecord': see https://developers.google.com/earth-engine/guides/tfrecord#formatoptions*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.image.toDrive

Creates a batch task to export an Image as a raster to Drive. Tasks can be started from the Tasks tab. "crsTransform", "scale", and "dimensions" are mutually exclusive.

Usage	Returns
`Export.image.toDrive(image, description, folder, fileNamePrefix, dimensions, region, scale, crs, crsTransform, maxPixels, shardSize, fileDimensions, skipEmptyTiles, fileFormat, formatOptions, priority)`	

Argument	Type	Details
`image`	Image	The image to export.
`description`	*String, optional*	*A human-readable name of the task. May contain letters, numbers, -, _ (no spaces). Defaults to "myExportImageTask".*
`folder`	*String, optional*	*The Google Drive Folder that the export will reside in. Note: (a) if the folder name exists at any level, the output is written to it, (b) if duplicate folder names exist, output is written to the most recently modified folder, (c) if the folder name does not exist, a new folder will be created at the root, and (d) folder names with separators (e.g. 'path/to/file') are interpreted as literal strings, not system paths. Defaults to Drive root.*
`fileNamePrefix`	*String, optional*	*The filename prefix. May contain letters, numbers, -, _ (no spaces). Defaults to the description.*
`dimensions`	*Number|String, optional*	*The dimensions to use for the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.*
`region`	*Geometry.LinearRing|Geometry.Polygon|String, optional*	*A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string.*
`scale`	*Number, optional*	*Resolution in meters per pixel. Defaults to 1000.*
`crs`	*String, optional*	*CRS to use for the exported image.*
`crsTransform`	*List, optional*	*Affine transform to use for the exported image. Requires "crs" to be defined.*
`maxPixels`	*Number, optional*	*Restrict the number of pixels in the export. By default, you will see an error if the export exceeds 1e8 pixels. Setting this value explicitly allows one to raise or lower this limit.*
`shardSize`	*Number, optional*	*Size in pixels of the tiles in which this image will be computed. Defaults to 256.*

Argument	Type	Details
`fileDimensions`	*List, optional*	*The dimensions in pixels of each image file, if the image is too large to fit in a single file. May specify a single number to indicate a square shape, or an array of two dimensions to indicate (width,height). Note that the image will still be clipped to the overall image dimensions. Must be a multiple of shardSize.*
`skipEmptyTiles`	*Boolean, optional*	*If true, skip writing empty (i.e. fully-masked) image tiles. Defaults to false. Only supported on GeoTIFF exports.*
`fileFormat`	*String, optional*	*The string file format to which the image is exported. Currently only 'GeoTIFF' and 'TFRecord' are supported, defaults to 'GeoTIFF'.*
`formatOptions`	*ImageExportFormatConfig, optional*	*A dictionary of string keys to format-specific options. For 'GeoTIFF': 'cloudOptimized' (Boolean), 'noData' (float). For 'TFRecord': see https://developers.google.com/earth-engine/guides/tfrecord#formatoptions*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.map.toCloudStorage

Creates a batch task to export an Image as a rectangular pyramid of map tiles for use with web map viewers. The map tiles will be accompanied by a reference index.html file that displays them using the Google Maps API, and an earth.html file for opening the map on Google Earth.

Usage	Returns
`Export.map.toCloudStorage(image, description, bucket, fileFormat, path, writePublicTiles, maxZoom, scale, minZoom, region, skipEmptyTiles, mapsApiKey, bucketCorsUris, priority)`	

Argument	Type	Details
`image`	Image	The image to export as tiles.
`description`	*String, optional*	*A human-readable name of the task. Defaults to "myExportMapTask".*
`bucket`	*String, optional*	*The destination bucket to write to.*
`fileFormat`	*String, optional*	*The map tiles' file format, one of "auto", "png", or "jpg". Defaults to "auto", which means that opaque tiles will be*

Argument	Type	Details
		encoded as "jpg" and tiles with transparency will be encoded as "png".
`path`	*String, optional*	*The string used as the output's path. A trailing "/" is optional. Defaults to the task's description.*
`writePublicTiles`	*Boolean, optional*	*Whether to write public tiles instead of using the bucket's default object ACL. Defaults to true and requires invoker to be OWNER of bucket.*
`maxZoom`	*Number, optional*	*The maximum zoom level of the map tiles to export.*
`scale`	*Number, optional*	*The max image resolution in meters per pixel, as an alternative to "maxZoom". The scale will be converted to the most appropriate maximum zoom level at the equator.*
`minZoom`	*Number, optional*	*The optional minimum zoom level of the map tiles to export. Defaults to zero.*
`region`	*Geometry.LinearRing|Geometry.Polygon|String, optional*	*A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string. Map tiles will be produced in the rectangular region containing this geometry.*
`skipEmptyTiles`	*Boolean, optional*	*If true, skip writing empty (i.e. fully-transparent) map tiles. Defaults to false. Only supported on GeoTIFF exports.*
`mapsApiKey`	*String, optional*	*Used in index.html to initialize the Google Maps API. This removes the "development purposes only" message from the map.*
`bucketCorsUris`	*List, optional*	*A list of domains (e.g. https://code.earthengine.google.com) that are allowed to retrieve the exported tiles from JavaScript. Setting the tiles to public is not enough to allow them to be accessible by a web page, so you must explicitly give domains access to the bucket. This is known as Cross-Origin-Resource-Sharing, or CORS. You can allow all domains to have access using "*", but this is generally discouraged. See https://cloud.google.com/storage/docs/cross-origin for more details.*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.table.toAsset

Creates a batch task to export a feature collection to an Earth Engine table asset. Tasks can be started from the Tasks tab.

Usage	Returns
`Export.table.toAsset(collection, description, assetId, maxVertices, priority)`	

Argument	Type	Details
`collection`	FeatureCollection	The feature collection to export.
`description`	*String, optional*	*A human-readable name of the task. Defaults to "myExportTableTask".*
`assetId`	*String, optional*	*The destination asset ID.*
`maxVertices`	*Number, optional*	*Max number of uncut vertices per geometry; geometries with more vertices will be cut into pieces smaller than this size.*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.table.toBigQuery

Creates a batch task to export a FeatureCollection to BigQuery. Tasks can be started from the Tasks tab.

Note that this feature is in Preview, and the API and behavior may change significantly. For more information, see https://developers.google.com/earth-engine/guides/export_to_bigquery

Usage	Returns
`Export.table.toBigQuery(collection, description, table, overwrite, append, selectors, maxVertices, priority)`	

Argument	Type	Details
`collection`	FeatureCollection	The feature collection to export.
`description`	*String, optional*	*A human-readable name of the task. Defaults to "myExportTableTask".*
`table`	*String, optional*	*The fully-qualifed BigQuery destination table in the following format: "project_id.dataset_id.table_id".*
`overwrite`	*Boolean, optional*	*[Not yet supported.] Whether the existing table should be overwritten by the result of this export. Defaults to false. The `overwrite` and `append` parameters cannot be `true` simultaneously. The export fails if the table already exists and both `overwrite` and `append` are `false`.*
`append`	*Boolean, optional*	*Whether table data should be appended if the table already exists and has a compatible schema. Defaults to false. The `overwrite` and `append` parameters cannot be `true` simultaneously. The export fails if the table already exists and both `overwrite` and `append` are `false`.*

Argument	Type	Details
`selectors`	*List, optional*	*A list of properties to include in the export; either a single string with comma-separated names or a list of strings.*
`maxVertices`	*Number, optional*	*Max number of uncut vertices per geometry; geometries with more vertices will be cut into pieces smaller than this size.*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.table.toCloudStorage

Creates a batch task to export a FeatureCollection as a table to Google Cloud Storage. Tasks can be started from the Tasks tab.

Usage	Returns
`Export.table.toCloudStorage(collection, description, bucket, fileNamePrefix, fileFormat, selectors, maxVertices, priority)`	

Argument	Type	Details
`collection`	FeatureCollection	The feature collection to export.
`description`	*String, optional*	*A human-readable name of the task. Defaults to "myExportTableTask".*
`bucket`	*String, optional*	*The Cloud Storage destination bucket.*
`fileNamePrefix`	*String, optional*	*The string used as the output's prefix. A trailing "/" indicates a path. Defaults to the description.*
`fileFormat`	*String, optional*	*The output format: "CSV" (default), "GeoJSON", "KML", "KMZ", "SHP", or "TFRecord".*
`selectors`	*List, optional*	*A list of properties to include in the export; either a single string with comma-separated names or a list of strings.*
`maxVertices`	*Number, optional*	*Max number of uncut vertices per geometry; geometries with more vertices will be cut into pieces smaller than this size.*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.table.toDrive

Creates a batch task to export a FeatureCollection as a table to Drive. Tasks can be started from the Tasks tab.

Usage	Returns
`Export.table.toDrive(collection, description, folder, fileNamePrefix, fileFormat, selectors, maxVertices, priority)`	

Argument	Type	Details
`collection`	FeatureCollection	The feature collection to export.
`description`	*String, optional*	*A human-readable name of the task. May contain letters, numbers, -, _ (no spaces). Defaults to "myExportTableTask".*
`folder`	*String, optional*	*The Google Drive Folder that the export will reside in. Note: (a) if the folder name exists at any level, the output is written to it, (b) if duplicate folder names exist, output is written to the most recently modified folder, (c) if the folder name does not exist, a new folder will be created at the root, and (d) folder names with separators (e.g. 'path/to/file') are interpreted as literal strings, not system paths. Defaults to Drive root.*
`fileNamePrefix`	*String, optional*	*The filename prefix. May contain letters, numbers, -, _ (no spaces). Defaults to the description.*
`fileFormat`	*String, optional*	*The output format: "CSV" (default), "GeoJSON", "KML", "KMZ", or "SHP", or "TFRecord".*
`selectors`	*List, optional*	*A list of properties to include in the export; either a single string with comma-separated names or a list of strings.*
`maxVertices`	*Number, optional*	*Max number of uncut vertices per geometry; geometries with more vertices will be cut into pieces smaller than this size.*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.table.toFeatureView

Creates a batch task to export a FeatureCollection to a FeatureView asset. Tasks can be started from the Tasks tab.

Usage	Returns
`Export.table.toFeatureView(collection, description, assetId, maxFeaturesPerTile, thinningStrategy, thinningRanking, zOrderRanking, priority)`	

Argument	Type	Details
`collection`	FeatureCollection	The feature collection to export.
`description`	*String, optional*	*A human-readable name of the task. May contain letters, numbers, -, _ (no spaces). Defaults to "myExportTableTask".*

Argument	Type	Details
`assetId`	*String, optional*	*The destination asset ID. May contain letters, numbers, -, _, and / (no spaces).*
`maxFeaturesPerTile`	*Number, optional*	*The max number of features that can intersect a tile. Can be a value between 0 and 2000; defaults to 500. Warning: Setting the max number of features to a value higher than 1000 may result in dropped tiles.*
`thinningStrategy`	*String, optional*	*The thinning strategy to use. Can either be HIGHER_DENSITY or GLOBALLY_CONSISTENT. Defaults to HIGHER_DENSITY. When thinning at a particular level of detail on the map, a higher density thinning strategy means that it tries to come as close as possible to the maxFeaturesPerTile limit for each tile. Globally-consistent thinning means that if a feature is removed by thinning, then all other features with equal or worse thinning rank will also be removed.*
`thinningRanking`	*List, optional*	*Comma-separated ranking rules defining the priority of how features should be thinned on the map. Defaults to ".minZoomLevel ASC". Each rule should be defined by a rule type and a direction (ASC or DESC), separated by a space. Valid rule types are: ".geometryType", ".minZoomLevel", or a feature property name. The value ".geometryType" refers to points, lines, and polygons. The value ".minZoomLevel" refers to the minimum zoom level that a feature is visible. Points are visible at all zoom levels, so they have the smallest minZoomLevel. For example, a valid set of ranking rules could be: 'my-property DESC, .geometryType ASC, .minZoomLevel ASC'. The same set of rules expressed as a list of strings would be: ['my-property DESC', '.geometryType ASC', '.minZoomLevel ASC']. This means when thinning at a particular level of detail on the map, prioritize features with a larger "my-property" value first (thin features with a smaller value of "my-property"), prioritize features with a smaller geometry type (e.g. thin out polygons before lines and thinning out lines before points), and prioritize features with a smaller minimum zoom level (points over large polygons over smaller polygons).*
`zOrderRanking`	*List, optional*	*Comma-separated ranking rules defining the z-order (stack order) of features displayed on the map. Defaults to ".minZoomLevel ASC". Uses the same format as thinningRanking. Each rule should be defined by a rule type and a direction (ASC or DESC), separated by a space. Valid rule types are: ".geometryType", ".minZoomLevel", or a feature property name. The value ".geometryType" refers to points, lines, and polygons. The value ".minZoomLevel" refers to the minimum zoom level that a feature is visible. Points are visible at all zoom levels, so they have the smallest minZoomLevel. For example, a valid set of ranking rules could be: 'my-property DESC, .geometryType ASC, .minZoomLevel ASC'. The same set of rules expressed as a list of strings would be: ['my-property DESC', '.geometryType ASC', '.minZoomLevel ASC']. This means when determining z-order of features at a particular level of detail on the map, features with a larger "my-property" value appear under features with a smaller value, features with a smaller geometry type appear under features with a larger geometry type (e.g. points under lines and lines under polygons), and features with a smaller min zoom level (larger features) appear under features with a larger min zoom level (smaller features).*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.video.toCloudStorage

Creates a batch task to export an ImageCollection as a video to Google Cloud Storage. The collection must only contain RGB images. Tasks can be started from the Tasks tab. "crsTransform", "scale", and "dimensions" are mutually exclusive.

Usage	Returns
`Export.video.toCloudStorage(collection, description, bucket, fileNamePrefix, framesPerSecond, dimensions, region, scale, crs, crsTransform, maxPixels, maxFrames, priority)`	

Argument	Type	Details
`collection`	ImageCollection	The image collection to export.
`description`	*String, optional*	*A human-readable name of the task. Defaults to "myExportVideoTask".*
`bucket`	*String, optional*	*The Cloud Storage destination bucket.*
`fileNamePrefix`	*String, optional*	*The string used as the output's prefix. A trailing "/" indicates a path. Defaults to the description.*
`framesPerSecond`	*Number, optional*	*The framerate of the exported video. Must be a value between 0.1 and 100. Defaults to 1.*
`dimensions`	*Number|String, optional*	*The dimensions to use for the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.*
`region`	*Geometry.LinearRing|Geometry.Polygon|String, optional*	*A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string.*
`scale`	*Number, optional*	*Resolution in meters per pixel.*
`crs`	*String, optional*	*CRS to use for the exported image. Defaults to the Google Maps Mercator projection, SR-ORG:6627.*
`crsTransform`	*String, optional*	*Affine transform to use for the exported image. Requires "crs" to be defined.*
`maxPixels`	*Number, optional*	*Restrict the number of pixels in the export. By default, you will see an error if the export exceeds 1e8 pixels. Setting this value explicitly allows one to raise or lower this limit.*
`maxFrames`	*Number, optional*	*Set the maximum number of frames to export. By default, a maximum of 1000 frames may be exported. By setting this explicitly, you may raise or lower this limit.*

Argument	Type	Details
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Export.video.toDrive

Creates a batch task to export an ImageCollection as a video to Drive. The collection must only contain RGB images. Tasks can be started from the Tasks tab. "crsTransform", "scale", and "dimensions" are mutually exclusive.

Usage	Returns
`Export.video.toDrive(collection, description, folder, fileNamePrefix, framesPerSecond, dimensions, region, scale, crs, crsTransform, maxPixels, maxFrames, priority)`	

Argument	Type	Details
`collection`	ImageCollection	The image collection to export.
`description`	*String, optional*	*A human-readable name of the task. May contain letters, numbers, -, _ (no spaces). Defaults to "myExportVideoTask".*
`folder`	*String, optional*	*The Google Drive Folder that the export will reside in. Note: (a) if the folder name exists at any level, the output is written to it, (b) if duplicate folder names exist, output is written to the most recently modified folder, (c) if the folder name does not exist, a new folder will be created at the root, and (d) folder names with separators (e.g. 'path/to/file') are interpreted as literal strings, not system paths. Defaults to Drive root.*
`fileNamePrefix`	*String, optional*	*The filename prefix. May contain letters, numbers, -, _ (no spaces). Defaults to the description.*
`framesPerSecond`	*Number, optional*	*The framerate of the exported video. Must be a value between 0.1 and 100. Defaults to 1.*
`dimensions`	*Number|String, optional*	*The dimensions to use for the exported image. Takes either a single positive integer as the maximum dimension or "WIDTHxHEIGHT" where WIDTH and HEIGHT are each positive integers.*
`region`	*Geometry.LinearRing|Geometry.Polygon|String, optional*	*A LinearRing, Polygon, or coordinates representing region to export. These may be specified as the Geometry objects or coordinates serialized as a string.*

Argument	Type	Details
`scale`	*Number, optional*	*Resolution in meters per pixel.*
`crs`	*String, optional*	*CRS to use for the exported image. Defaults to the Google Maps Mercator projection, SR-ORG:6627.*
`crsTransform`	*String, optional*	*Affine transform to use for the exported image. Requires "crs" to be defined.*
`maxPixels`	*Number, optional*	*Restrict the number of pixels in the export. By default, you will see an error if the export exceeds 1e8 pixels. Setting this value explicitly allows one to raise or lower this limit.*
`maxFrames`	*Number, optional*	*Set the maximum number of frames to export. By default, a maximum of 1000 frames may be exported. By setting this explicitly, you may raise or lower this limit.*
`priority`	*Number, optional*	*The priority of the task within the project. Higher priority tasks are scheduled sooner. Must be an integer between 0 and 9999. Defaults to 100.*

Map.add

Adds an item to the map. Can also be used to add widgets like ui.Label as well as some non-widget objects like ui.Map.Layer.

Returns the map.

Usage	Returns
`Map.add(item)`	ui.Map

Argument	Type	Details
`item`	Object	The item to add.

Map.addLayer

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

Returns the new map layer.

Usage	Returns
`Map.addLayer(eeObject, visParams, name, shown, opacity)`	ui.Map.Layer

Argument	Type	Details
`eeObject`	Collection|Feature|Image|RawMapId	The object to add to the map.
`visParams`	*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 CSS 3.0 color string or a hex string in "RRGGBB" format. Ignored when eeObject is a map ID.*
`name`	*String, optional*	*The name of the layer. Defaults to "Layer N".*
`shown`	*Boolean, optional*	*A flag indicating whether the layer should be on by default.*
`opacity`	*Number, optional*	*The layer's opacity represented as a number between 0 and 1. Defaults to 1.*

Map.centerObject

Centers the map view on a given object.

Caution: providing a large or complex collection as input can result in poor performance. Collating the geometry of collections does not scale well; use the smallest collection (or geometry) that is required to achieve the desired outcome.

Returns the map.

Usage	Returns
`Map.centerObject(object, zoom, onComplete)`	ui.Map

Argument	Type	Details
`object`	Element|Geometry	An object to center on - a geometry, image or feature.
`zoom`	*Number, optional*	*The zoom level, from 0 to 24. If unspecified, computed based on the object's bounding box.*
`onComplete`	*Function, optional*	*A callback which is triggered after the recentering completes successfully. Passing this parameter causes the `centerObject` operation to run asynchronously.*

Map.clear

Clears the map by removing all layers, listeners, and widgets and restoring the options to their defaults.

Returns the map.

Usage	Returns
`Map.clear()`	ui.Map

No arguments.

Map.drawingTools

Returns the Map's drawing tools, which can be used to create and edit shapes on the map.

Usage	Returns
`Map.drawingTools()`	ui.Map.DrawingTools

No arguments.

Map.getBounds

Returns the bounds of the current map view, as a list in the format [west, south, east, north] in degrees.

Usage	Returns
`Map.getBounds(`*`asGeoJSON`*`)`	GeoJSONGeometry|List|String

Argument	Type	Details
`asGeoJSON`	*Boolean, optional*	*If true, returns map bounds as GeoJSON.*

Map.getCenter

Returns the coordinates at the center of the map.

Usage	Returns
`Map.getCenter()`	Geometry.Point

No arguments.

Map.getScale

Returns the approximate pixel scale of the current map view, in meters.

Usage	Returns
`Map.getScale()`	Number|String

No arguments.

Map.getZoom

Returns the current zoom level of the map.

Usage	Returns
`Map.getZoom()`	Number

No arguments.

Map.layers

Returns the list of layers associated with the default map.

Usage	Returns
`Map.layers()`	ui.data.ActiveList

No arguments.

Map.onChangeBounds

Registers a callback that's fired when the map bounds change. This is fired during pan, zoom, and when the map's bounds are changed programmatically. Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onChangeBounds(callback)`	String

Argument	Type	Details
`callback`	Function	The callback to fire when the map bounds change. The callback is passed two parameters: an object containing the coordinates of the new map center (with keys lon, lat, and zoom) and the map widget itself.

Map.onChangeCenter

Registers a callback that's fired when the map center changes. This is fired during pan or when the map's center is changed programmatically.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onChangeCenter(callback)`	String

Argument	Type	Details
`callback`	Function	The callback to fire when the map center changes. The callback is passed two parameters: an object containing the coordinates of the new center (with keys lon and lat) and the map widget itself.

Map.onChangeZoom

Registers a callback that's fired when the map zoom level changes.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onChangeZoom(callback)`	String

Argument	Type	Details
`callback`	Function	The callback to fire when the map zoom change. The callback is passed two parameters: the new zoom level and the map widget itself.

Map.onClick

Registers a callback that's fired when the map is clicked.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onClick(callback)`	String

Argument	Type	Details
`callback`	Function	The callback to fire when the map is clicked. The callback is passed an object containing the coordinates of the clicked point (with keys lon and lat) and the map widget.

Map.onIdle

Registers a callback that's fired when the map stops moving.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onIdle(callback)`	String

Argument	Type	Details
`callback`	Function	The callback to fire when the map becomes idle. The callback is passed two parameters: an object containing the coordinates of the map center (with keys lon, lat, and zoom) and the map widget itself.

Map.onTileLoaded

Registers a callback that's fired when a map tile has been loaded.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onTileLoaded(callback)`	String

Argument	Type	Details
`callback`	Function	Called with an array of per layer values. Each value is the fraction of tiles still pending: a value of 0 means there are no more tiles to load for the layer.

Map.remove

Removes the given item from the map, if it exists.

Returns the removed item or null if it hadn't been added to the map.

Usage	Returns
`Map.remove(item)`	Object

Argument	Type	Details
`item`	Object	The item to remove.

Map.setCenter

Centers the map view at a given coordinates with the given zoom level.

Returns the map.

Usage	Returns
`Map.setCenter(lon, lat, zoom)`	ui.Map

Argument	Type	Details
`lon`	Number	The longitude of the center, in degrees.
`lat`	Number	The latitude of the center, in degrees.
`zoom`	*Number, optional*	*The zoom level, from 0 to 24.*

Map.setControlVisibility

Sets the visibility of the controls on the map.

Returns this ui.Map.

Usage	Returns
`Map.setControlVisibility(`*`all`*`, `*`layerList`*`, `*`zoomControl`*`, `*`scaleControl`*`, `*`mapTypeControl`*`, `*`fullscreenControl`*`, `*`drawingToolsControl`*`)`	ui.Map

Argument	Type	Details
`all`	*Boolean, optional*	*Whether to show all controls. False hides all controls; true shows all controls. Overridden by individually set parameters. Note that setting this explicitly will affect any additional controls added in the future.*
`layerList`	*Boolean, optional*	*When false, hides the layer list panel or, when true, allows the layer list panel's visibility to be determined by the presence of layers in the list. The default is to show the list.*
`zoomControl`	*Boolean, optional*	*Whether the zoom control is visible. Defaults to true.*
`scaleControl`	*Boolean, optional*	*Whether to show the control which indicates the scale at the map's current zoom level. Defaults to true.*
`mapTypeControl`	*Boolean, optional*	*Whether to show the control that allows the user to change the base map. Defaults to true.*
`fullscreenControl`	*Boolean, optional*	*Whether to show the control that allows the user to make the map full-screen. Defaults to true.*
`drawingToolsControl`	*Boolean, optional*	*Whether to show the control that allows the user to add or edit the geometry drawing tools. Defaults to true.*

Map.setGestureHandling

Controls how gestures are handled on the map.

See https://developers.google.com/maps/documentation/javascript/reference/map#MapOptions.gestureHandling.

Usage	Returns
`Map.setGestureHandling(option)`	

Argument	Type	Details
`option`	String	The option that controls how gestures are handled on the map. Allowed values: • "cooperative": Scroll events and one-finger touch gestures scroll the page, and do not zoom or pan the map. Two-finger touch gestures pan and zoom the map. Scroll events with a ctrl key or ⌘ key pressed zoom the map. In this mode the map cooperates with the page. • "greedy": All touch gestures and scroll events pan or zoom the map. • "none": The map cannot be panned or zoomed by user gestures. • "auto": (default) Gesture handling is either cooperative or greedy, depending on whether the page is scrollable or in an iframe.

Map.setLocked

Limits panning and zooming on the map.

- To lock both panning and zooming, set locked to true and nothing else.

- To allow panning and limit the min and max zoom, set locked to false and supply the minZoom and maxZoom parameters.

- To disallow panning and limit min and max zoom, set locked to true and supply the minZoom and maxZoom parameters.

- To reset the map to default, set locked to false and nothing else.

Usage	Returns
`Map.setLocked(locked, minZoom, maxZoom)`	

Argument	Type	Details
`locked`	Boolean	Whether the map should be locked or not.
`minZoom`	*Number, optional*	*(optional) The minimum zoom for the map, between 0 and 24, inclusive.*
`maxZoom`	*Number, optional*	*(optional) The maximum zoom for the map, between 0 and 24, inclusive.*

Map.setOptions

Modifies the Google Maps basemap. Allows for:

1) Setting the current MapType. 2) Providing custom styles for the basemap (MapTypeStyles). 3) Setting the list of available mapTypesIds for the basemap.

If called with no parameters, resets the map type to the google default.

Returns the map.

Usage	Returns
`Map.setOptions(`*`mapTypeId`*`, `*`styles`*`, `*`types`*`)`	ui.Map

Argument	Type	Details
`mapTypeId`	*String, optional*	*A mapTypeId to set the basemap to. Can be one of "ROADMAP", "SATELLITE", "HYBRID" or "TERRAIN" to select one of the standard Google Maps API map types, or one of the keys specified in the opt_styles dictionary. If left as null and only 1 style is specified in opt_styles, that style will be used.*
`styles`	*Object, optional*	*A dictionary of custom MapTypeStyle objects keyed with a name that will appear in the map's Map Type Controls. See: https://developers.google.com/maps/documentation/javascript/reference#MapTypeStyle*
`types`	*List, optional*	*A list of mapTypeIds to make available. If omitted, but opt_styles is specified, appends all of the style keys to the standard Google Maps API map types.*

Map.setZoom

Sets the zoom level of the map.

Returns this ui.Map.

Usage	Returns
`Map.setZoom(zoom)`	ui.Map

Argument	Type	Details
`zoom`	Number	The zoom level, from 0 to 24, to set for the map.

Map.style

Returns the Map's style ActiveDictionary, which can be modified to update the Map's styles.

In addition to the standard UI API styles listed in the ui.Panel.style() documentation, the Map supports the following custom style option:

- cursor, which can be 'crosshair' or 'hand' (default)

Usage	Returns
`Map.style()`	ui.data.ActiveDictionary

No arguments.

Map.unlisten

Deletes callbacks.

Usage	Returns
`Map.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
`idOrType`	*String, optional*	*Either an ID returned by listen() when a callback was registered, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks registered with that event type are deleted. If nothing is passed, all callbacks are deleted.*

Map.widgets

Returns the list of the widgets currently on the map.

Usage	Returns
`Map.widgets()`	ui.data.ActiveList

No arguments.

exports

The reserved namespace for exporting objects as module members.

Usage	Returns
`exports()`	

No arguments.

print

Prints the arguments to the console.

Usage	Returns
`print(var_args)`	

Argument	Type	Details
`var_args`	VarArgs	The objects to print.

require

Retrieves the script found at a given path as a module. The module is used to access exposed members of the required script.

Returns returns an object that represents exported members from the required module.

Usage	Returns
`require(path)`	Object

Argument	Type	Details
`path`	String	The path to the script to include as a module. Paths must be absolute, such as: "users/homeFolder/repo:path/to/file".

ui.Button

A clickable button with a text label.

Usage	Returns
`ui.Button(`*`label`*`, `*`onClick`*`, `*`disabled`*`, `*`style`*`, `*`imageUrl`*`)`	ui.Button

Argument	Type	Details
`label`	*String, optional*	*The button's label. Defaults to an empty string.*

Argument	Type	Details
`onClick`	*Function, optional*	*A callback fired when the button is clicked. The callback is passed the button widget.*
`disabled`	*Boolean, optional*	*Whether the button is disabled. Defaults to false.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this widget. Defaults to an empty object.*
`imageUrl`	*String, optional*	*Optional image url. If provided, the button will be rendered as an image and the value text will be shown on mouse hover. Only data: urls and icons loaded from gstatic.com are allowed.*

ui.Button.getDisabled

Returns whether the button is disabled.

Usage	Returns
`Button.getDisabled()`	Boolean

Argument	Type	Details
this: `ui.button`	ui.Button	The ui.Button instance.

ui.Button.getImageUrl

Returns the url of the image if it exists.

Usage	Returns
`Button.getImageUrl()`	String

Argument	Type	Details
this: `ui.button`	ui.Button	The ui.Button instance.

ui.Button.getLabel

Returns the button's label.

Usage	Returns
`Button.getLabel()`	String

Argument	Type	Details
this: `ui.button`	ui.Button	The ui.Button instance.

ui.Button.onClick

Registers a callback that's fired when the button is clicked.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Button.onClick(callback)`	String

Argument	Type	Details
this: `ui.button`	ui.Button	The ui.Button instance.
`callback`	Function	The callback to fire when the button is clicked. The callback is passed the button widget.

ui.Button.setDisabled

Sets whether the button is disabled.

Returns this button.

Usage	Returns
`Button.setDisabled(disabled)`	ui.Button

Argument	Type	Details
this: `ui.button`	ui.Button	The ui.Button instance.
`disabled`	Boolean	Whether the button is disabled.

ui.Button.setImageUrl

Shows the button as image, which will render instead of the label text.

Returns this button.

Usage	Returns
`Button.setImageUrl(imageUrl)`	ui.Button

Argument	Type	Details
this: `ui.button`	ui.Button	The ui.Button instance.
`imageUrl`	String	The url of the image.

ui.Button.setLabel

Sets the button's label.

Returns this button.

Usage	Returns
`Button.setLabel(label)`	ui.Button

Argument	Type	Details
this: `ui.button`	ui.Button	The ui.Button instance.
`label`	String	The button's label.

ui.Button.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')

- width, maxWidth, minWidth (e.g. '100px')

- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')

- color, backgroundColor (e.g. 'red' or '#FF0000')

- border (e.g. '1px solid black')

- fontSize (e.g. '24px')

- fontStyle (e.g. 'italic')

- fontWeight (e.g. 'bold' or '100')

- fontFamily (e.g. 'monospace' or 'serif')

- textAlign (e.g. 'left' or 'center')

- textDecoration (e.g. 'underline' or 'line-through')

- whiteSpace (e.g. 'nowrap' or 'pre')

- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')

- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Button.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Button.unlisten

Deletes callbacks.

Usage	Returns
`Button.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

Argument	Type	Details
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.Chart

A chart widget.

Usage	Returns
`ui.Chart(`*`dataTable`*`, `*`chartType`*`, `*`options`*`, `*`view`*`, `*`downloadable`*`)`	ui.Chart

Argument	Type	Details
`dataTable`	*List, optional*	*A 2-D array of data or a Google Visualization DataTable literal. See: http://developers.google.com/chart/interactive/docs/reference#DataTable*
`chartType`	*String, optional*	*The chart type; e.g 'ScatterChart', 'LineChart', and 'ColumnChart'. For the complete list of charts, see: https://developers.google.com/chart/interactive/docs/gallery*
`options`	*Object, optional*	*An object defining chart style options such as:* `title` (string) The title of the chart. `colors` (Array) An array of colors used to draw the chart. *Its format should follow the Google Visualization API's options: https://developers.google.com/chart/interactive/docs/customizing_charts*
`view`	*Object, optional*	*Sets a DataView initializer object, which acts as a filter over the underlying data. See: https://developers.google.com/chart/interactive/docs/reference#DataView*
`downloadable`	*Boolean, optional*	*Whether the chart can be downloaded as CSV, SVG, and PNG. Defaults to true.*

ui.Chart.array.values

Generates a Chart from an array. Plots separate series for each 1-D vector along the given axis.

- X-axis = Array index along axis, optionally labeled by xLabels.

- Y-axis = Value.

- Series = Vector, described by indices of the non-axis array axes.

Returns a chart.

Usage	Returns
`ui.Chart.array.values(array, axis, xLabels)`	ui.Chart

Argument	Type	Details
`array`	Array|List	Array to chart.
`axis`	Number	The axis along which to generate the 1-D vector series.
`xLabels`	*Array|List, optional*	*Labels for ticks along the x-axis of the chart.*

ui.Chart.feature.byFeature

Generates a Chart from a set of features. Plots the value of one or more properties for each feature:

- X-axis = Features labeled by xProperty (default: 'system:index').

- Y-axis = Values of yProperties (default: all properties).

- Series = Names of yProperties.

The values are ordered along the x-axis in the same order as the input features.

Returns a chart.

Usage	Returns
`ui.Chart.feature.byFeature(features, xProperty, yProperties)`	ui.Chart

Argument	Type	Details
`features`	Feature|FeatureCollection|List	The features to include in the chart.
`xProperty`	*String, optional*	*The property used as the value of each feature on the x-axis. Defaults to 'system:index'.*
`yProperties`	*List, optional*	*Property or properties used on the y-axis. If omitted, all properties of all features will be charted on the y-axis (except xProperty).*

ui.Chart.feature.byProperty

Generates a Chart from a set of features. Plots property values of one or more features.

- X-axis = Property name, labeled by xProperties (default: all properties).

- Y-axis = Property value (must be numeric).

- Series = Features, labeled by seriesProperty (default: 'system:index').

All properties except seriesProperty are included on the x-axis by default.

Returns a chart.

Usage	Returns
`ui.Chart.feature.byProperty(features, xProperties, seriesProperty)`	ui.Chart

Argument	Type	Details
`features`	Feature|FeatureCollection|List	The features to include in the chart.
`xProperties`	*List, optional*	*One of (1) a property to be plotted on the x-axis; (2) a list of properties to be plotted on the x-axis; or (3) a (property, label) dictionary specifying labels for properties to be used as values on the x-axis. If omitted, all properties will be plotted on the x-axis, labeled with their names.*
`seriesProperty`	*String, optional*	*The name of the property used to label each feature in the legend. Defaults to 'system:index'.*

ui.Chart.feature.groups

Generates a Chart from a set of features. Plots the value of a given property across groups of features. Features with the same value of groupProperty will be grouped and plotted as a single series.

- X-axis = xProperty values.

- Y-axis = yProperty values.

- Series = Feature groups, by seriesProperty.

Returns a chart.

Usage	Returns
`ui.Chart.feature.groups(features, xProperty, yProperty, seriesProperty)`	ui.Chart

Argument	Type	Details
`features`	Feature|FeatureCollection|List	The features to include in the chart.

Argument	Type	Details
`xProperty`	String	Property to be used as the label for each feature on the x-axis.
`yProperty`	String	Property to be plotted on the y-axis.
`seriesProperty`	String	Property used to determine feature groups. Features with the same value of groupProperty will be plotted as a single series on the chart.

ui.Chart.feature.histogram

Generates a Chart from a set of features. Computes and plots a histogram of the given property.

- X-axis = Histogram buckets (of property value).

- Y-axis = Frequency (i.e. the number of features whose value of property lands within the x-axis bucket bounds).

Returns a chart.

Usage	Returns
`ui.Chart.feature.histogram(features, property, maxBuckets, minBucketWidth, maxRaw)`	ui.Chart

Argument	Type	Details
`features`	Feature|FeatureCollection|List	The features to include in the chart.
`property`	String	The name of the property to generate the histogram for.
`maxBuckets`	*Number, optional*	*The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2. Not used when the value of property is non-numeric.*
`minBucketWidth`	*Number, optional*	*The minimum histogram bucket width, or null to allow any power of 2. Not used when property is non-numeric.*
`maxRaw`	*Number, optional*	*The number of values to accumulate before building the initial histogram. Not used when property is non-numeric.*

ui.Chart.getChartType

Returns this chart's type; e.g 'ScatterChart', 'LineChart', and 'ColumnChart'. For the complete list of charts, see: https://developers.google.com/chart/interactive/docs/gallery

Usage	Returns
`Chart.getChartType()`	String

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.

ui.Chart.getDataTable

Returns the DataTable containing data for this chart. See: http://developers.google.com/chart/interactive/docs/reference#DataTable

Usage	Returns
`Chart.getDataTable()`	Object

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.

ui.Chart.getDownloadable

Returns whether the chart can be downloaded as CSV, SVG, and PNG.

Usage	Returns
`Chart.getDownloadable()`	Boolean

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.

ui.Chart.getOptions

Returns this chart's options. See: https://developers.google.com/chart/interactive/docs/customizing_charts

Usage	Returns
`Chart.getOptions()`	Object

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.

ui.Chart.getView

Returns this chart's DataView initializer object, which acts as a filter over the underlying data in the chart. See: https://developers.google.com/chart/interactive/docs/reference#DataView

Usage	Returns
`Chart.getView()`	Object

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.

ui.Chart.image.byClass

Generates a Chart from an image. Plots derived band values in classified regions in an image.

- X-axis = Band name (all bands except the class band are charted).

- Y-axis = Band value.

- Series = Class label.

Returns a chart.

Usage	Returns
`ui.Chart.image.byClass(image, classBand, region, reducer, scale, classLabels, xLabels)`	ui.Chart

Argument	Type	Details
`image`	Image	Classified image to derive band values from.
`classBand`	Number|String	The class label band in this image.

Argument	Type	Details
`region`	*Feature|FeatureCollection|Geometry, optional*	*The region to reduce. If omitted, uses the entire image.*
`reducer`	*Reducer, optional*	*Reducer that generates the value(s) for the y-axis. Must return a single value per band. Defaults to ee.Reducer.mean().*
`scale`	*Number, optional*	*Scale to use with the reducer in meters.*
`classLabels`	*List., optional*	*A dictionary of labels used to identify classes in the series legend. If omitted, classes will be labeled with the value of classBand.*
`xLabels`	*List, optional*	*A list of labels used to label bands on the xAxis. Must have one fewer elements than the number of image bands. If omitted, bands will be labeled with their names. If the labels are numeric (e.g. wavelengths), x-axis will be continuous.*

ui.Chart.image.byRegion

Generates a Chart from an image. Extracts and plots band values in one or more regions in the image, with each band in a separate series.

- X-axis = Region labeled by xProperty (default: 'system:index')

- Y-axis = Reducer output.

- Series = Band name.

Returns a chart.

Usage	Returns
`ui.Chart.image.byRegion(image, regions, reducer, scale, xProperty)`	ui.Chart

Argument	Type	Details
`image`	Image	Image to extract band values from.
`regions`	*Feature|FeatureCollection|Geometry|List, optional*	*Regions to reduce. Defaults to the image's footprint.*
`reducer`	*Reducer, optional*	*Reducer that generates the value(s) for the y-axis. Must return a single value per band. Defaults to ee.Reducer.mean().*
`scale`	*Number, optional*	*Scale to use with the reducer in meters.*
`xProperty`	*String, optional*	*Property to be used as the label for each Region on the x-axis. Defaults to 'system:index'.*

ui.Chart.image.doySeries

Generates a Chart from an ImageCollection. Plots derived values of each band in a region for a each day of the year.

- X-axis: Day of year (startDay to endDay, defaults to 1 to 366).

- Y-axis: Derived band value (reduced within the region and across years).

- Series: Band names.

Returns a chart.

Usage	Returns
`ui.Chart.image.doySeries(imageCollection, region, regionReducer, scale, yearReducer, startDay, endDay)`	ui.Chart

Argument	Type	Details
`imageCollection`	ImageCollection	The ImageCollection to chart.
`region`	*Feature|FeatureCollection|Geometry, optional*	*The region to reduce. Defaults to the union of all geometries in the image collection.*
`regionReducer`	*Reducer, optional*	*Reducer for aggregating band values within the region. Must return a single value. Defaults to ee.Reducer.mean().*
`scale`	*Number, optional*	*Scale to use with the region reducer in meters.*
`yearReducer`	*Reducer, optional*	*Reducer for aggregating regionReducer outputs across years (for a given day). Must return a single value. Defaults to ee.Reducer.mean().*
`startDay`	*Number, optional*	*Day of year to start the series. Must be between 1 and 366.*
`endDay`	*Number, optional*	*Day of year to end the series. Must be between startDay and 366.*

ui.Chart.image.doySeriesByRegion

Generates a Chart from an ImageCollection. Plots the derived value of the given band in different regions at each day-of-year.

- X-axis: Day of year (startDay to endDay, defaults to 1 to 366).

- Y-axis: Derived band value (reduced within the region and across years).

- Series: Regions.

Returns a chart.

Usage	Returns
`ui.Chart.image.doySeriesByRegion(imageCollection, bandName, regions, regionReducer, scale, yearReducer, seriesProperty, startDay, endDay)`	ui.Chart

Argument	Type	Details
`imageCollection`	ImageCollection	The ImageCollection to chart.
`bandName`	Number|String	The name of the band to chart.
`regions`	Feature|FeatureCollection|Geometry|List	The regions to reduce.
`regionReducer`	*Reducer, optional*	*Reducer for aggregating band values within the region. Must return a single value. Defaults to ee.Reducer.mean().*
`scale`	*Number, optional*	*Scale to use with the region reducer in meters.*
`yearReducer`	*Reducer, optional*	*Reducer for aggregating band values across years (for a given day of year). Must return a single value. Defaults to ee.Reducer.mean().*
`seriesProperty`	*String, optional*	*Property of features in opt_regions to be used for series labels. Defaults to 'system:index'.*
`startDay`	*Number, optional*	*Day of year to start the series. Must be between 1 and 366.*
`endDay`	*Number, optional*	*Day of year to end the series. Must be between startDay and 366.*

ui.Chart.image.doySeriesByYear

Generates a Chart from an ImageCollection. Plots the derived value of the given band in a region for each day-of-year across different years.

- X-axis: Day of year (startDay to endDay, defaults to 1 to 366).

- Y-axis: Derived band value (reduced within the region).

- Series: Years.

Returns a chart.

Usage	Returns
`ui.Chart.image.doySeriesByYear(imageCollection, bandName, region, regionReducer, scale, sameDayReducer, startDay, endDay)`	ui.Chart

Argument	Type	Details
`imageCollection`	ImageCollection	The ImageCollection to chart.
`bandName`	Number|String	The name of the band to chart.
`region`	*Feature|FeatureCollection|Geometry, optional*	*The region to reduce. Defaults to the union of all geometries in the image collection.*
`regionReducer`	*Reducer, optional*	*Reducer for aggregating band values within the region. Must return a single value. Defaults to ee.Reducer.mean().*
`scale`	*Number, optional*	*Scale to use with the region reducer in meters.*
`sameDayReducer`	*Reducer, optional*	*Reducer for aggregating band values across images with the same (DoY, year) pair. Must return a single value. Defaults to ee.Reducer.mean().*
`startDay`	*Number, optional*	*Day of year to start the series. Must be between 1 and 366.*
`endDay`	*Number, optional*	*Day of year to end the series. Must be between startDay and 366.*

ui.Chart.image.histogram

Generates a Chart from an image. Computes and plots histograms of the values of the bands in the specified region of the image.

- X-axis: Histogram buckets (of band value).

- Y-axis: Frequency (number of pixels with a band value in the bucket).

Returns a chart.

Usage	Returns
`ui.Chart.image.histogram(image, region, scale, maxBuckets, minBucketWidth, maxRaw, maxPixels)`	ui.Chart

Argument	Type	Details
`image`	Image	The image to generate a histogram from.
`region`	*Feature|FeatureCollection|Geometry, optional*	*The region to reduce. If omitted, uses the entire image.*
`scale`	*Number, optional*	*The pixel scale used when applying the histogram reducer, in meters.*

Argument	Type	Details
`maxBuckets`	*Number, optional*	*The maximum number of buckets to use when building a histogram; will be rounded up to a power of 2.*
`minBucketWidth`	*Number, optional*	*The minimum histogram bucket width, or null to allow any power of 2.*
`maxRaw`	*Number, optional*	*The number of values to accumulate before building the initial histogram.*
`maxPixels`	*Number, optional*	*If specified, overrides the maximum number of pixels allowed in the histogram reduction. Defaults to 1e6.*

ui.Chart.image.regions

Generates a Chart from an image. Extracts and plots the value of each band in one or more regions.

- X-axis = Band labeled by xProperty (default: band name).

- Y-axis = Reducer output.

- Series = Region labeled by seriesProperty (default: 'system:index').

Returns a chart.

Usage	Returns
`ui.Chart.image.regions(image, regions, reducer, scale, seriesProperty, xLabels)`	ui.Chart

Argument	Type	Details
`image`	Image	Image to extract band values from.
`regions`	*Feature|FeatureCollection|Geometry|List, optional*	*Regions to reduce. Defaults to the image's footprint.*
`reducer`	*Reducer, optional*	*Reducer that generates the value(s) for the y-axis. Must return a single value per band.*
`scale`	*Number, optional*	*The pixel scale in meters.*
`seriesProperty`	*String, optional*	*Property to be used as the label for each region in the legend. Defaults to 'system:index'.*
`xLabels`	*List, optional*	*A list of labels used for bands on the x-axis. Must have the same number of elements as the image bands. If omitted, bands will be labeled with their names. If the labels are numeric (e.g. wavelengths), x-axis will be continuous.*

ui.Chart.image.series

Generates a Chart from an ImageCollection. Plots derived values of each band in a region across images. Usually a time series.

- X-axis: Image, labeled by xProperty value.

- Y-axis: Band value.

- Series: Band names.

Returns a chart.

Usage	Returns
`ui.Chart.image.series(imageCollection, region, reducer, scale, xProperty)`	ui.Chart

Argument	Type	Details
`imageCollection`	ImageCollection	An ImageCollection with data to be included in the chart.
`region`	Feature|FeatureCollection|Geometry	The region to reduce.
`reducer`	*Reducer, optional*	*Reducer that generates the values for the y-axis. Must return a single value. Defaults to ee.Reducer.mean().*
`scale`	*Number, optional*	*Scale to use with the reducer in meters.*
`xProperty`	*String, optional*	*Property to be used as the label for each image on the x-axis. Defaults to 'system:time_start'.*

ui.Chart.image.seriesByRegion

Generates a Chart from an image collection. Extracts and plots the value of the specified band in each region for each image in the collection. Usually a time series.

- X-axis = Image labeled by xProperty (default: 'system:time_start').

- Y-axis = Reducer output.

- Series = Region labeled by seriesProperty (default: 'system:index').

Returns a chart.

Usage	Returns
`ui.Chart.image.seriesByRegion(imageCollection, regions, reducer, band, scale, xProperty, seriesProperty)`	ui.Chart

Argument	Type	Details
`imageCollection`	ImageCollection	An ImageCollection with data to be included in the chart.
`regions`	Feature|FeatureCollection|Geometry|List	The regions to reduce.
`reducer`	Reducer	Reducer that generates the value for the y-axis. Must return a single value.
`band`	*Number|String, optional*	*The band name to reduce using the reducer. Defaults to the first band.*
`scale`	*Number, optional*	*Scale to use with the reducer in meters.*
`xProperty`	*String, optional*	*Property to be used as the label for each image on the x-axis. Defaults to 'system:time_start'.*
`seriesProperty`	*String, optional*	*Property of features in opt_regions to be used for series labels. Defaults to 'system:index'.*

ui.Chart.onClick

Registers a callback that's fired when the chart is clicked.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Chart.onClick(callback)`	String

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.
`callback`	Function	The callback to fire when the chart is clicked. The callback is passed three arguments: the x-value, the y-value, and the series name. Time values are represented in UTC epoch milliseconds, like "system:time_start" values on assets. If the user clicks on a legend entry to select an entire series, the x- and y-values are null. If the user clicks an already-selected point, all arguments are null, indicating the selection was cleared.

ui.Chart.setChartType

Sets the chartType of this chart.

Returns this chart.

Usage	Returns
`Chart.setChartType(chartType)`	ui.Chart

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.
`chartType`	String	The chart type; e.g 'ScatterChart', 'LineChart', and 'ColumnChart'. For the complete list of charts, see: https://developers.google.com/chart/interactive/docs/gallery

ui.Chart.setDataTable

Sets the DataTable containing data for this chart.

Returns this chart.

Usage	Returns
`Chart.setDataTable(dataTable)`	ui.Chart

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.
`dataTable`	List	A 2-D array of data to chart or a Google Visualization DataTable literal. See: http://developers.google.com/chart/interactive/docs/reference#DataTable

ui.Chart.setDownloadable

Sets a view for this chart.

Returns this chart.

Usage	Returns
`Chart.setDownloadable(Whether)`	ui.Chart

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.
`Whether`	Boolean	the chart can be downloaded as CSV, SVG, and PNG.

ui.Chart.setOptions

Sets options used to style this chart.

Returns this chart.

Usage	Returns
`Chart.setOptions(options)`	ui.Chart

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.
`options`	Object	An object defining chart style options such as: `title` (string) The title of the chart. `colors` (Array) An array of colors used to draw the chart. Its format should follow the Google Visualization API's options: https://developers.google.com/chart/interactive/docs/customizing_charts

ui.Chart.setSeriesNames

Returns a copy of this chart with updated series names.

Usage	Returns
`Chart.setSeriesNames(seriesNames, `*`seriesIndex`*`)`	ui.Chart

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.
`seriesNames`	Dictionary|Dictionary	New series names. If it's a string, the name of the series at seriesIndex is set to seriesNames. If it's a list, the value at index i in the list is used as a label for series number i. If it's a dictionary or an object, it's treated as a map from existing series names to new series names. In the last two cases, seriesIndex is ignored.
`seriesIndex`	*Number, optional*	*The index of the series to rename. Ignored if seriesNames is a list or dictionary. Series are 0-indexed.*

ui.Chart.setView

Sets a view for this chart.

Returns this chart.

Usage	Returns
`Chart.setView(view)`	ui.Chart

Argument	Type	Details
this: `ui.chart`	ui.Chart	The ui.Chart instance.
`view`	Object	A DataView initializer object, which acts as a filter over the underlying data in the chart. See: https://developers.google.com/chart/interactive/docs/reference#DataView

ui.Chart.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')

- width, maxWidth, minWidth (e.g. '100px')

- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')

- color, backgroundColor (e.g. 'red' or '#FF0000')

- border (e.g. '1px solid black')

- fontSize (e.g. '24px')

- fontStyle (e.g. 'italic')

- fontWeight (e.g. 'bold' or '100')

- fontFamily (e.g. 'monospace' or 'serif')

- textAlign (e.g. 'left' or 'center')

- textDecoration (e.g. 'underline' or 'line-through')

- whiteSpace (e.g. 'nowrap' or 'pre')

- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')

- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Chart.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Chart.unlisten

Deletes callbacks.

Usage	Returns
`Chart.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

Argument	Type	Details
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.Checkbox

A checkbox with a label.

Usage	Returns
`ui.Checkbox(`*`label`*`, `*`value`*`, `*`onChange`*`, `*`disabled`*`, `*`style`*`)`	ui.Checkbox

Argument	Type	Details
`label`	*String, optional*	*The checkbox's label. Defaults to an empty string.*
`value`	*Boolean, optional*	*Whether the checkbox is checked. A null value indicates that the checkbox is in an indeterminate state. Defaults to false.*
`onChange`	*Function, optional*	*A callback to fire when the value of the checkbox changes. The callback is passed a boolean indicating whether the checkbox is now checked and the checkbox widget.*
`disabled`	*Boolean, optional*	*Whether the checkbox is disabled. Defaults to false.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this widget. See style() documentation.*

ui.Checkbox.getDisabled

Returns whether the checkbox is disabled.

Usage	Returns
`Checkbox.getDisabled()`	Boolean

Argument	Type	Details
this: `ui.checkbox`	ui.Checkbox	The ui.Checkbox instance.

ui.Checkbox.getLabel

Returns the checkbox's label.

Usage	Returns
`Checkbox.getLabel()`	String

Argument	Type	Details
this: `ui.checkbox`	ui.Checkbox	The ui.Checkbox instance.

ui.Checkbox.getValue

Returns whether the checkbox is checked. A null value indicates the checkbox is in an indeterminate state.

Usage	Returns
`Checkbox.getValue()`	Boolean

Argument	Type	Details
this: `ui.checkbox`	ui.Checkbox	The ui.Checkbox instance.

ui.Checkbox.onChange

Registers a callback that's fired when the value of the checkbox changes.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Checkbox.onChange(callback)`	String

Argument	Type	Details
this: `ui.checkbox`	ui.Checkbox	The ui.Checkbox instance.
`callback`	Function	The callback to fire when the value of the checkbox changes. The callback is passed a boolean indicating whether the checkbox is now checked and the checkbox widget.

ui.Checkbox.setDisabled

Sets whether the checkbox is disabled.

Returns this checkbox.

Usage	Returns
`Checkbox.setDisabled(disabled)`	ui.Checkbox

Argument	Type	Details
this: `ui.checkbox`	ui.Checkbox	The ui.Checkbox instance.
`disabled`	Boolean	Whether the checkbox is disabled.

ui.Checkbox.setLabel

Sets the checkbox's label.

Returns this checkbox.

Usage	Returns
`Checkbox.setLabel(value)`	ui.Checkbox

Argument	Type	Details
this: `ui.checkbox`	ui.Checkbox	The ui.Checkbox instance.
`value`	String	The new label for the checkbox.

ui.Checkbox.setValue

Sets whether the checkbox is checked.

Returns this checkbox.

Usage	Returns
`Checkbox.setValue(value, `*`trigger`*`)`	ui.Checkbox

Argument	Type	Details
this: `ui.checkbox`	ui.Checkbox	The ui.Checkbox instance.
`value`	Boolean	Whether the checkbox is checked. A null value indicates the checkbox is in an indeterminate state.
`trigger`	*Boolean, optional*	*Whether to trigger onChange callbacks when the checked property changes. Defaults to true.*

ui.Checkbox.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')

- width, maxWidth, minWidth (e.g. '100px')

- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')

- color, backgroundColor (e.g. 'red' or '#FF0000')

- border (e.g. '1px solid black')

- fontSize (e.g. '24px')

- fontStyle (e.g. 'italic')

- fontWeight (e.g. 'bold' or '100')

- fontFamily (e.g. 'monospace' or 'serif')

- textAlign (e.g. 'left' or 'center')

- textDecoration (e.g. 'underline' or 'line-through')

- whiteSpace (e.g. 'nowrap' or 'pre')

- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')

- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Checkbox.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Checkbox.unlisten

Deletes callbacks.

Usage	Returns
`Checkbox.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.DateSlider

A draggable target that ranges linearly between two dates. The date slider can be configured to display dates of various interval sizes, including day, 8-day, and year. The value of the slider is displayed as a label alongside it.

Usage	Returns
`ui.DateSlider(`*`start`*`, `*`end`*`, `*`value`*`, `*`period`*`, `*`onChange`*`, `*`disabled`*`, `*`style`*`)`	ui.DateSlider

Argument	Type	Details
`start`	*Date|Number|String, optional*	*The start date, as a UTC timestamp, date string, or ee.Date. Defaults to one week ago.*
`end`	*Date|Number|String, optional*	*The end date, as a UTC timestamp, date string, or ee.Date. Defaults to today.*

Argument	Type	Details
`value`	*Date|Number|String, optional*	*The initial value. The value is an array consisting of the start and end date for the selected date range, but for convenience, it can be set by specifying the start date alone. Defaults to yesterday.*
`period`	*Number, optional*	*The interval size for values on the slider in days. Defaults to one.*
`onChange`	*Function, optional*	*A callback to fire when the slider's state changes. The callback is passed an ee.DateRange representing the slider's current value and the slider widget.*
`disabled`	*Boolean, optional*	*Whether the slider is disabled. Defaults to false.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this widget. Defaults to an empty object.*

ui.DateSlider.getDisabled

Returns whether the slider is disabled.

Usage	Returns
`DateSlider.getDisabled()`	Boolean

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.

ui.DateSlider.getEnd

Returns the slider's end date as a UTC timestamp.

Usage	Returns
`DateSlider.getEnd()`	Number

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.

ui.DateSlider.getPeriod

Returns the slider's period interval.

Usage	Returns
`DateSlider.getPeriod()`	Number

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.

ui.DateSlider.getStart

Returns the slider's start date as a UTC timestamp.

Usage	Returns
`DateSlider.getStart()`	Number

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.

ui.DateSlider.getValue

Returns the slider's current value, and array with the start and end datetimes as epoch UTC timestamps.

Usage	Returns
`DateSlider.getValue()`	List

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.

ui.DateSlider.onChange

Registers a callback that's fired when the slider's value changes.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DateSlider.onChange(callback)`	String

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.
`callback`	Function	The callback to fire when the slider's state changes. The callback is passed an ee.DateRange representing the slider's current value and the slider widget.

ui.DateSlider.setDisabled

Sets whether the slider is disabled.

Returns this slider.

Usage	Returns
`DateSlider.setDisabled(disabled)`	ui.DateSlider

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.
`disabled`	Boolean	Whether the slider is disabled.

ui.DateSlider.setEnd

Sets the end date of the slider.

Returns this slider.

Usage	Returns
`DateSlider.setEnd(value)`	ui.DateSlider

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.

Argument	Type	Details
`value`	Number|String	The slider's end date.

ui.DateSlider.setPeriod

Sets the period interval of the slider.

Returns this slider.

Usage	Returns
`DateSlider.setPeriod(value)`	ui.DateSlider

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.
`value`	Number	The slider's period interval.

ui.DateSlider.setStart

Sets the start date of the slider.

Returns this slider.

Usage	Returns
`DateSlider.setStart(start)`	ui.DateSlider

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.
`start`	Number|String	The start date. Defaults to one week ago.

ui.DateSlider.setValue

Set the value of the slider.

Returns this slider.

Usage	Returns
`DateSlider.setValue(value, trigger)`	ui.DateSlider

Argument	Type	Details
this: `ui.dateslider`	ui.DateSlider	The ui.DateSlider instance.
`value`	Number|String	The value to set on the slider.
`trigger`	*Boolean, optional*	*Whether to trigger onChange callbacks when the value property changes. Defaults to true.*

ui.DateSlider.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')

- width, maxWidth, minWidth (e.g. '100px')

- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')

- color, backgroundColor (e.g. 'red' or '#FF0000')

- border (e.g. '1px solid black')

- fontSize (e.g. '24px')

- fontStyle (e.g. 'italic')

- fontWeight (e.g. 'bold' or '100')

- fontFamily (e.g. 'monospace' or 'serif')

- textAlign (e.g. 'left' or 'center')

- textDecoration (e.g. 'underline' or 'line-through')

- whiteSpace (e.g. 'nowrap' or 'pre')

- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')

- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`DateSlider.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.DateSlider.unlisten

Deletes callbacks.

Usage	Returns
`DateSlider.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.Label

A text label.

Usage	Returns
`ui.Label(`*`value`*`, `*`style`*`, `*`targetUrl`*`, `*`imageUrl`*`)`	ui.Label

Argument	Type	Details
`value`	*String, optional*	*The text to display. Defaults to an empty string.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this widget. See style() documentation.*

Argument	Type	Details
`targetUrl`	*String, optional*	*The url to link to. Defaults to an empty string.*
`imageUrl`	*String, optional*	*Optional image url. If provided, the label will be rendered as an image and the value text will be shown on mouse hover. Only data: urls and icons loaded from gstatic.com are allowed.*

ui.Label.getImageUrl

Returns the url of the image if it exists.

Usage	Returns
`Label.getImageUrl()`	String

Argument	Type	Details
this: `ui.label`	ui.Label	The ui.Label instance.

ui.Label.getUrl

Returns the url of the label if it exists.

Usage	Returns
`Label.getUrl()`	String

Argument	Type	Details
this: `ui.label`	ui.Label	The ui.Label instance.

ui.Label.getValue

Returns the value of the label.

Usage	Returns
`Label.getValue()`	String

Argument	Type	Details
this: `ui.label`	ui.Label	The ui.Label instance.

ui.Label.setImageUrl

Sets the label to an image, which will render instead of the value text.

Returns this label.

Usage	Returns
`Label.setImageUrl(imageUrl)`	ui.Label

Argument	Type	Details
this: `ui.label`	ui.Label	The ui.Label instance.
`imageUrl`	String	The url of the image.

ui.Label.setUrl

Sets the url of the label, which will cause it to render as a link.

Returns this label.

Usage	Returns
`Label.setUrl(targetUrl)`	ui.Label

Argument	Type	Details
this: `ui.label`	ui.Label	The ui.Label instance.
`targetUrl`	String	The url of the hyperlink.

ui.Label.setValue

Sets the value of the label.

Returns this label.

Usage	Returns
`Label.setValue(value)`	ui.Label

Argument	Type	Details
this: `ui.label`	ui.Label	The ui.Label instance.
`value`	String	The value of the label.

ui.Label.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')
- width, maxWidth, minWidth (e.g. '100px')
- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')
- color, backgroundColor (e.g. 'red' or '#FF0000')
- border (e.g. '1px solid black')
- fontSize (e.g. '24px')
- fontStyle (e.g. 'italic')
- fontWeight (e.g. 'bold' or '100')
- fontFamily (e.g. 'monospace' or 'serif')
- textAlign (e.g. 'left' or 'center')
- textDecoration (e.g. 'underline' or 'line-through')
- whiteSpace (e.g. 'nowrap' or 'pre')
- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')
- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Label.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Map

A Google map.

Usage	Returns
`ui.Map(`*`center`*`, `*`onClick`*`, `*`style`*`)`	ui.Map

Argument	Type	Details
`center`	*Object, optional*	*An object containing the latitude ('lat'), longitude ('lon') and optionally the zoom level ('zoom') for the map.*
`onClick`	*Function, optional*	*A callback fired when the map is clicked. The callback is passed an object containing the coordinates of the clicked point on the map (with keys lon and lat) and the map widget itself.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this map. See style() documentation.*

ui.Map.CloudStorageLayer

A layer generated from Cloud Storage tiles for display on a ui.Map.

Usage	Returns
`ui.Map.CloudStorageLayer(bucket, path, maxZoom, `*`suffix`*`, `*`name`*`, `*`shown`*`, `*`opacity`*`)`	ui.Map.CloudStorageLayer

Argument	Type	Details
`bucket`	String	The bucket that contains the tiles.
`path`	String	The path to this layer's tiles, relative to the bucket. A trailing "/" is optional.
`maxZoom`	Number	The maximum zoom level for which there are tiles.

Argument	Type	Details
`suffix`	*String, optional*	*The tile source file suffix, if any.*
`name`	*String, optional*	*The name of the layer.*
`shown`	*Boolean, optional*	*Whether the layer is initially shown. Defaults to true.*
`opacity`	*Number, optional*	*The layer's opacity represented as a number between 0 and 1. Defaults to 1.*

ui.Map.CloudStorageLayer.getBucket

Returns the name of this layer's bucket.

Usage	Returns
`CloudStorageLayer.getBucket()`	String

Argument	Type	Details
this: `ui.map.cloudstoragelayer`	ui.Map.CloudStorageLayer	The ui.Map.CloudStorageLayer instance.

ui.Map.CloudStorageLayer.getMaxZoom

Returns the maximum zoom level of this layer's tileset.

Usage	Returns
`CloudStorageLayer.getMaxZoom()`	Number

Argument	Type	Details
this: `ui.map.cloudstoragelayer`	ui.Map.CloudStorageLayer	The ui.Map.CloudStorageLayer instance.

ui.Map.CloudStorageLayer.getName

Returns the name of the layer.

Usage	Returns
`CloudStorageLayer.getName()`	String

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.CloudStorageLayer.getOpacity

Returns the layer's opacity represented as a number between 0 and 1.

Usage	Returns
`CloudStorageLayer.getOpacity()`	Number

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.CloudStorageLayer.getPath

Returns the path within the bucket to the tiles.

Usage	Returns
`CloudStorageLayer.getPath()`	String

Argument	Type	Details
this: `ui.map.cloudstoragelayer`	ui.Map.CloudStorageLayer	The ui.Map.CloudStorageLayer instance.

ui.Map.CloudStorageLayer.getShown

Returns whether the layer is shown.

Usage	Returns
`CloudStorageLayer.getShown()`	Boolean

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.CloudStorageLayer.getSuffix

Returns the suffix for this layer's tile files.

Usage	Returns
`CloudStorageLayer.getSuffix()`	String

Argument	Type	Details
this: `ui.map.cloudstoragelayer`	ui.Map.CloudStorageLayer	The ui.Map.CloudStorageLayer instance.

ui.Map.CloudStorageLayer.setBucket

Sets the bucket for this layer.

Returns this map layer.

Usage	Returns
`CloudStorageLayer.setBucket(bucket)`	ui.Map.CloudStorageLayer

Argument	Type	Details
this: `ui.map.cloudstoragelayer`	ui.Map.CloudStorageLayer	The ui.Map.CloudStorageLayer instance.
`bucket`	String	The name of the Cloud Storage bucket with this layer's tiles.

ui.Map.CloudStorageLayer.setMaxZoom

Sets the maximum zoom level for tiles. When the user zooms in beyond this level, the parent tile at this level will be fetched and zoomed on the client.

Returns this map layer.

Usage	Returns
`CloudStorageLayer.setMaxZoom(maxZoom)`	ui.Map.CloudStorageLayer

Argument	Type	Details
this: `ui.map.cloudstoragelayer`	ui.Map.CloudStorageLayer	The ui.Map.CloudStorageLayer instance.
`maxZoom`	Number	The maximum zoom level with tiles.

ui.Map.CloudStorageLayer.setName

Sets the name of the layer.

Returns this map layer.

Usage	Returns
`CloudStorageLayer.setName(`*`name`*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`name`	*String, optional*	*The name of the layer.*

ui.Map.CloudStorageLayer.setOpacity

Sets the opacity of the layer.

Returns this map layer.

Usage	Returns
`CloudStorageLayer.setOpacity(`*`opacity`*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`opacity`	*Number, optional*	*The layer's opacity represented as a number between 0 and 1.*

ui.Map.CloudStorageLayer.setPath

Sets the location of the folder from which the layer will retrieve its tiles.

Returns this map layer.

Usage	Returns
`CloudStorageLayer.setPath(path)`	ui.Map.CloudStorageLayer

Argument	Type	Details
this: `ui.map.cloudstoragelayer`	ui.Map.CloudStorageLayer	The ui.Map.CloudStorageLayer instance.
`path`	String	The path to this layer's tiles, relative to the bucket.

ui.Map.CloudStorageLayer.setShown

Sets the visibility of the layer.

Returns this map layer.

Usage	Returns
`CloudStorageLayer.setShown(`*shown*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`shown`	*Boolean, optional*	*Whether the layer is shown.*

ui.Map.CloudStorageLayer.setSuffix

Sets the CloudStorageLayer's file suffix.

Returns this map layer.

Usage	Returns
`CloudStorageLayer.setSuffix(suffix)`	ui.Map.CloudStorageLayer

Argument	Type	Details
this: `ui.map.cloudstoragelayer`	ui.Map.CloudStorageLayer	The ui.Map.CloudStorageLayer instance.
`suffix`	String	The suffix for the tile files, for example ".png".

ui.Map.DrawingTools

A set of tools for drawing on a map.

Usage	Returns
`ui.Map.DrawingTools(`*`layers`*`, `*`shape`*`, `*`selected`*`, `*`shown`*`, `*`linked`*`)`	ui.Map.DrawingTools

Argument	Type	Details
`layers`	*List, optional*	*An array of geometry layers with which to initialize the drawing tools.*
`shape`	*String, optional*	*The shape to draw. One of the following: point, line, polygon, or rectangle. Defaults to polygon.*
`selected`	*ui.Map.GeometryLayer, optional*	*The selected geometry layer. Defaults to null.*
`shown`	*Boolean, optional*	*When false, hides the drawing tools or, when true, shows the shape selecter and allows the list panel's visibility to be determined by the presence of geometry layers in the list. Defaults to true.*
`linked`	*Boolean, optional*	*Whether the drawing tools are linked to the geometries in the imports pane. When false, the tools do not display imported geometries. Defaults to false.*

ui.Map.DrawingTools.addLayer

Adds a given list of ee.Geometry objects to the drawing tools as a geometry layer.

Returns the new geometry layer.

Usage	Returns
`DrawingTools.addLayer(geometries, name, color, shown, locked)`	ui.Map.GeometryLayer

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`geometries`	List	The geometries with which to initialize the layer.
`name`	*String, optional*	*The name of the layer.*
`color`	*String, optional*	*The CSS color of shapes in the layer, for instance "white" or "#FFFFFF".*
`shown`	*Boolean, optional*	*Whether to show the shapes in the layer. Defaults to true.*
`locked`	*Boolean, optional*	*Whether to lock shape editing in the layer. Defaults to false.*

ui.Map.DrawingTools.clear

Clears the drawing tools.

Returns this set of drawing tools.

Usage	Returns
`DrawingTools.clear()`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.draw

Enters drawing mode, in which a click on the map will begin drawing the selected shape.

Returns this set of drawing tools.

Usage	Returns
`DrawingTools.draw()`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.edit

Starts editing the selected layer.

Returns this set of drawing tools.

Usage	Returns
`DrawingTools.edit()`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.get

Returns either a clone of this object or, if a key is provided, the value of the property with the passed-in key. Look at the constructor's parameters to see which properties are available.

Usage	Returns
`DrawingTools.get(`*`key`*`)`	Object

Argument	Type	Details
this: `ui.data.activedictionary`	ui.data.ActiveDictionary	The ui.data.ActiveDictionary instance.
`key`	*String, optional*	*The key of the property to retrieve.*

ui.Map.DrawingTools.getDrawModes

Gets the available draw modes on the drawing tool. The available draw mode shapes are: point, line, polygon, and rectangle.

Returns the list of enabled draw modes.

Usage	Returns
`DrawingTools.getDrawModes()`	List

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.getLinked

Returns whether the drawing tools' geometries are linked to those in the imports panel.

Usage	Returns
`DrawingTools.getLinked()`	Boolean

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.getMap

Returns the map for these drawing tools or null if the drawing tools have not been added to a map.

Usage	Returns
`DrawingTools.getMap()`	ui.Map

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.getSelected

Returns the selected layer.

Usage	Returns
`DrawingTools.getSelected()`	ui.Map.GeometryLayer

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.getShape

Returns the shape drawn when in drawing mode.

Usage	Returns
`DrawingTools.getShape()`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.getShown

Returns whether the drawing tools are shown.

Usage	Returns
`DrawingTools.getShown()`	Boolean

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.layers

Returns the list of geometry layers in the drawing tools.

Usage	Returns
`DrawingTools.layers()`	ui.data.ActiveList

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.onDraw

Registers a callback that's fired when a shape is drawn.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onDraw(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`callback`	Function	The callback to fire when a shape is drawn. The callback is passed three parameters: the added ee.Geometry, the GeometryLayer to which the geometry was added, and the ui.Map.DrawingTools widget that the event listener is bound to.

ui.Map.DrawingTools.onEdit

Registers a callback that's fired when a shape is edited.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onEdit(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

Argument	Type	Details
`callback`	Function	The callback to fire when a shape is edited. The callback is passed three parameters: the edited ee.Geometry, the GeometryLayer to which the edited geometry belongs, and the ui.Map.DrawingTools widget that the event listener is bound to.

ui.Map.DrawingTools.onErase

Registers a callback that's fired when a shape is erased.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onErase(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`callback`	Function	The callback to fire when a shape is erased. The callback is passed three parameters: the removed ee.Geometry, the GeometryLayer from which the geometry was removed, and the ui.Map.DrawingTools widget that the event listener is bound to.

ui.Map.DrawingTools.onLayerAdd

Registers a callback that's fired when a layer is added.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onLayerAdd(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`callback`	Function	The callback to fire when a layer is added. The callback is passed two parameters: the added GeometryLayer and the ui.Map.DrawingTools widget

Argument	Type	Details
		that the event listener is bound to.

ui.Map.DrawingTools.onLayerConfig

Registers a callback that's fired after a layer's name or color is changed.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onLayerConfig(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`callback`	Function	The callback to fire after a layer is configured. The callback is passed two parameters: the configured GeometryLayer and the ui.Map.DrawingTools widget that the event listener is bound to.

ui.Map.DrawingTools.onLayerRemove

Registers a callback that's fired when a layer is removed.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onLayerRemove(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`callback`	Function	The callback to fire when a layer is removed. The callback is passed two parameters: the removed GeometryLayer and the ui.Map.DrawingTools widget that the event listener is bound to.

ui.Map.DrawingTools.onLayerSelect

Registers a callback that's fired when a layer is selected.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onLayerSelect(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`callback`	Function	The callback to fire when a shape is selected. The callback is passed two parameters: the selected GeometryLayer (or null for deselect) and the ui.Map.DrawingTools widget that the event listener is bound to.

ui.Map.DrawingTools.onSelect

Registers a callback that's fired when a shape is selected.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onSelect(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`callback`	Function	The callback to fire when a shape is selected. The callback is passed three parameters: the selected ee.Geometry, the GeometryLayer to which the selected geometry belongs, and the ui.Map.DrawingTools widget that the event listener is bound to.

ui.Map.DrawingTools.onShapeChange

Registers a callback that's fired when a drawing mode shape is changed.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`DrawingTools.onShapeChange(callback)`	String

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`callback`	Function	The callback to fire when the shape is changed. The callback is passed two parameters: the drawing mode shape as a string (or null for cancel) and the ui.Map.DrawingTools widget that the event listener is bound to. The shape values are: • point • line • polygon • rectangle • null

ui.Map.DrawingTools.set

Sets the value of a given property. Throws an error if the key provided is not supported by the object. Look at the constructor's parameters to see which properties can be set.

Returns this ui.data.ActiveDictionary.

Usage	Returns
`DrawingTools.set(keyOrDict, value)`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.data.activedictionary`	ui.data.ActiveDictionary	The ui.data.ActiveDictionary instance.
`keyOrDict`	Object|String	Either the key of the property to set or a dictionary of key/value pairs to set on the object.
`value`	*Object, optional*	*The property's new value. This is required when the first argument is a key string.*

ui.Map.DrawingTools.setDrawModes

Sets the available draw mode shapes on the drawing tool. The available draw mode shapes are: point, line, polygon, and rectangle.

Usage	Returns
`DrawingTools.setDrawModes(`*`drawModes`*`)`	

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`drawModes`	*List, optional*	*The list of draw modes to enable. Defaults to all supported ones.*

ui.Map.DrawingTools.setLinked

Sets whether the drawing tools' geometries are linked to the imports panel or isolated to the map.

Returns these ui.Map.DrawingTools.

Usage	Returns
`DrawingTools.setLinked(linked)`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`linked`	Boolean	Whether the geometries should be linked to the imports panel. When false, all geometries are local to the map instance.

ui.Map.DrawingTools.setSelected

Sets the selected layer.

Returns this set of drawing tools.

Usage	Returns
`DrawingTools.setSelected(`*`layer`*`)`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`layer`	*ui.Map.GeometryLayer, optional*	*The layer to select or null to deselect all layers.*

ui.Map.DrawingTools.setShape

Sets the draw mode shape and starts draw mode. The available draw mode shapes are: point, line, polygon, and rectangle.

Returns this set of drawing tools.

Usage	Returns
`DrawingTools.setShape(shape)`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`shape`	String	The shape to draw.

ui.Map.DrawingTools.setShown

Sets the visibility of the shape selector and geometry layer list.

Returns this set of drawing tools.

Usage	Returns
`DrawingTools.setShown(shown)`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`shown`	Boolean	Whether to show the drawing tools.

ui.Map.DrawingTools.stop

Closes the drawing tools, exiting interactive drawing or editing.

Returns this set of drawing tools.

Usage	Returns
`DrawingTools.stop()`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

ui.Map.DrawingTools.toFeatureCollection

Returns a feature collection in which each geometry in the drawing tools is a feature.

Usage	Returns
`DrawingTools.toFeatureCollection(indexProperty)`	FeatureCollection

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.
`indexProperty`	String	A property with this name will be assigned to every feature in the returned collection. The value of the property will be a number that corresponds to the index of the geometry layer to which the geometry belongs.

ui.Map.DrawingTools.unlisten

Deletes callbacks.

Usage	Returns
`DrawingTools.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.map.drawingtools`	ui.Map.DrawingTools	The ui.Map.DrawingTools instance.

Argument	Type	Details
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.Map.FeatureViewLayer

A layer generated from a FeatureView asset for display on a ui.Map.

Usage	Returns
`ui.Map.FeatureViewLayer(assetId, visParams, name, shown, opacity)`	ui.Map.FeatureViewLayer

Argument	Type	Details
`assetId`	String	The asset ID for the FeatureView.
`visParams`	*Object, optional*	*The visualization parameters for this layer.*
`name`	*String, optional*	*The name of the layer, which appears in the list of layers and when inspecting this layer. Defaults to the asset ID.*
`shown`	*Boolean, optional*	*Whether the layer is initially shown on the map. Defaults to true.*
`opacity`	*Number, optional*	*The layer's opacity represented as a number between 0 and 1. Defaults to 1.*

ui.Map.FeatureViewLayer.getAssetId

Returns the asset ID for the FeatureView asset backing this layer.

Usage	Returns
`FeatureViewLayer.getAssetId()`	String

Argument	Type	Details
this: `ui.map.featureviewlayer`	ui.Map.FeatureViewLayer	The ui.Map.FeatureViewLayer instance.

ui.Map.FeatureViewLayer.getName

Returns the name of the layer.

Usage	Returns
`FeatureViewLayer.getName()`	String

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.FeatureViewLayer.getOpacity

Returns the layer's opacity represented as a number between 0 and 1.

Usage	Returns
`FeatureViewLayer.getOpacity()`	Number

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.FeatureViewLayer.getShown

Returns whether the layer is shown.

Usage	Returns
`FeatureViewLayer.getShown()`	Boolean

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.FeatureViewLayer.getVisParams

Returns the visualization parameters for this layer.

Usage	Returns
`FeatureViewLayer.getVisParams()`	Object

Argument	Type	Details
this: `ui.map.featureviewlayer`	ui.Map.FeatureViewLayer	The ui.Map.FeatureViewLayer instance.

ui.Map.FeatureViewLayer.setAssetId

Changes the asset being displayed on this layer.

Returns this map layer.

Usage	Returns
`FeatureViewLayer.setAssetId(assetId)`	ui.Map.FeatureViewLayer

Argument	Type	Details
this: `ui.map.featureviewlayer`	ui.Map.FeatureViewLayer	The ui.Map.FeatureViewLayer instance.
`assetId`	String	The asset ID for the FeatureView backing this layer.

ui.Map.FeatureViewLayer.setName

Sets the name of the layer.

Returns this map layer.

Usage	Returns
`FeatureViewLayer.setName(`*`name`*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`name`	*String, optional*	*The name of the layer.*

ui.Map.FeatureViewLayer.setOpacity

Sets the opacity of the layer.

Returns this map layer.

Usage	Returns
`FeatureViewLayer.setOpacity(`*`opacity`*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`opacity`	*Number, optional*	*The layer's opacity represented as a number between 0 and 1.*

ui.Map.FeatureViewLayer.setShown

Sets the visibility of the layer.

Returns this map layer.

Usage	Returns
`FeatureViewLayer.setShown(`*`shown`*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`shown`	*Boolean, optional*	*Whether the layer is shown.*

ui.Map.FeatureViewLayer.setVisParams

Sets the visualization parameters for this layer.

Returns this map layer.

Usage	Returns
`FeatureViewLayer.setVisParams(`*`visParams`*`)`	ui.Map.FeatureViewLayer

Argument	Type	Details
this: `ui.map.featureviewlayer`	ui.Map.FeatureViewLayer	The ui.Map.FeatureViewLayer instance.
`visParams`	*Object, optional*	*The visualization parameters for this layer.*

ui.Map.GeometryLayer

A layer of ee.Geometries for display as shapes on a ui.Map.

Usage	Returns
`ui.Map.GeometryLayer(`*`geometries`*`, `*`name`*`, `*`color`*`, `*`shown`*`, `*`locked`*`)`	ui.Map.GeometryLayer

Argument	Type	Details
`geometries`	*List, optional*	*The geometries with which to initialize the layer.*
`name`	*String, optional*	*The name of the layer.*
`color`	*String, optional*	*The CSS color of shapes in the layer, for instance "white" or "#FFFFFF". Defaults to "#000000" (black).*
`shown`	*Boolean, optional*	*Whether to show the shapes in the layer. Defaults to true.*
`locked`	*Boolean, optional*	*Whether to lock shape editing in the layer. Defaults to false.*

ui.Map.GeometryLayer.fromGeometry

Resets the layer's geometries by parsing individual geometries from an ee.Geometry.

Returns this geometry layer.

Usage	Returns
`GeometryLayer.fromGeometry(geometry)`	ui.Map.GeometryLayer

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.
`geometry`	Geometry	A geometry with which to reset the layer's geometries.

ui.Map.GeometryLayer.geometries

Returns the active list of geometries associated with the layer.

Usage	Returns
`GeometryLayer.geometries()`	ui.data.ActiveList

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.

ui.Map.GeometryLayer.get

Returns either a clone of this object or, if a key is provided, the value of the property with the passed-in key. Look at the constructor's parameters to see which properties are available.

Usage	Returns
`GeometryLayer.get(`*`key`*`)`	Object

Argument	Type	Details
this: `ui.data.activedictionary`	ui.data.ActiveDictionary	The ui.data.ActiveDictionary instance.
`key`	*String, optional*	*The key of the property to retrieve.*

ui.Map.GeometryLayer.getColor

Returns the color of the layer.

Usage	Returns
`GeometryLayer.getColor()`	String

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.

ui.Map.GeometryLayer.getEeObject

Returns the EE object associated with the layer.

Usage	Returns
`GeometryLayer.getEeObject()`	Feature|FeatureCollection|Geometry

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.

ui.Map.GeometryLayer.getLocked

Returns whether the shapes in the layer are shown.

Usage	Returns
`GeometryLayer.getLocked()`	Boolean

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.

ui.Map.GeometryLayer.getName

Returns the name of the layer.

Usage	Returns
`GeometryLayer.getName()`	String

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.

ui.Map.GeometryLayer.getShown

Returns whether the shapes in the layer are shown.

Usage	Returns
`GeometryLayer.getShown()`	Boolean

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.

ui.Map.GeometryLayer.openConfigurationDialog

Opens a configuration dialog for the layer. Use onLayerConfig to register a callback for when the user makes changes using the dialog.

Returns the geometry layer to be updated by the dialog.

Usage	Returns
`GeometryLayer.openConfigurationDialog()`	ui.Map.GeometryLayer

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.

ui.Map.GeometryLayer.set

Sets the value of a given property. Throws an error if the key provided is not supported by the object. Look at the constructor's parameters to see which properties can be set.

Returns this ui.data.ActiveDictionary.

Usage	Returns
`GeometryLayer.set(keyOrDict, value)`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.data.activedictionary`	ui.data.ActiveDictionary	The ui.data.ActiveDictionary instance.
`keyOrDict`	Object|String	Either the key of the property to set or a dictionary of key/value pairs to set on the object.

Argument	Type	Details
`value`	*Object, optional*	*The property's new value. This is required when the first argument is a key string.*

ui.Map.GeometryLayer.setColor

Sets the CSS color of shapes in the layer.

Returns this map layer.

Usage	Returns
`GeometryLayer.setColor(color)`	ui.Map.GeometryLayer

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.
`color`	String	The color of the layer.

ui.Map.GeometryLayer.setLocked

Sets the locked state of the layer. A locked layer disallows adding, removing, or editing the geometries on the layer from the user interface.

Returns this map layer.

Usage	Returns
`GeometryLayer.setLocked(locked)`	ui.Map.GeometryLayer

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.
`locked`	Boolean	Whether the layer is locked.

ui.Map.GeometryLayer.setName

Sets the name of the layer.

Returns this map layer.

Usage	Returns
`GeometryLayer.setName(name)`	ui.Map.GeometryLayer

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.
`name`	String	The name of the layer.

ui.Map.GeometryLayer.setShown

Sets the visibility of shapes in the layer.

Returns this map layer.

Usage	Returns
`GeometryLayer.setShown(shown)`	ui.Map.GeometryLayer

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.
`shown`	Boolean	Whether the layer is shown.

ui.Map.GeometryLayer.toGeometry

Returns the layer's geometries as a single ee.Geometry.

Usage	Returns
`GeometryLayer.toGeometry()`	Geometry

Argument	Type	Details
this: `ui.map.geometrylayer`	ui.Map.GeometryLayer	The ui.Map.GeometryLayer instance.

ui.Map.Layer

A layer generated from an Earth Engine object for display on a ui.Map.

Usage	Returns
`ui.Map.Layer(`*`eeObject`*`, `*`visParams`*`, `*`name`*`, `*`shown`*`, `*`opacity`*`)`	ui.Map.Layer

Argument	Type	Details
`eeObject`	*Collection|Feature|Image, optional*	*The object to add to the map. Defaults to an empty ee.Image.*
`visParams`	*FeatureVisualizationParameters|ImageVisualizationParameters, optional*	*The visualization parameters. See ee.data.getMapId() docs.*
`name`	*String, optional*	*The name of the layer.*
`shown`	*Boolean, optional*	*Whether the layer is initially shown. Defaults to true.*
`opacity`	*Number, optional*	*The layer's opacity represented as a number between 0 and 1. Defaults to 1.*

ui.Map.Layer.getEeObject

Returns the layer's ee.Object.

Usage	Returns
`Layer.getEeObject()`	Collection|Feature|Image

Argument	Type	Details
this: `ui.map.layer`	ui.Map.Layer	The ui.Map.Layer instance.

ui.Map.Layer.getName

Returns the name of the layer.

Usage	Returns
`Layer.getName()`	String

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.Layer.getOpacity

Returns the layer's opacity represented as a number between 0 and 1.

Usage	Returns
`Layer.getOpacity()`	Number

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.Layer.getShown

Returns whether the layer is shown.

Usage	Returns
`Layer.getShown()`	Boolean

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.

ui.Map.Layer.getVisParams

Returns the layer's visualization parameters.

Usage	Returns
`Layer.getVisParams()`	FeatureVisualizationParameters|ImageVisualizationParameters

Argument	Type	Details
this: `ui.map.layer`	ui.Map.Layer	The ui.Map.Layer instance.

ui.Map.Layer.setEeObject

Sets the layer's ee.Object.

Returns this map layer.

Usage	Returns
`Layer.setEeObject(`*`eeObject`*`)`	ui.Map.Layer

Argument	Type	Details
this: `ui.map.layer`	ui.Map.Layer	The ui.Map.Layer instance.
`eeObject`	*Collection|Feature|Image, optional*	*The object to add to the map. Defaults to an empty ee.Image.*

ui.Map.Layer.setName

Sets the name of the layer.

Returns this map layer.

Usage	Returns
`Layer.setName(`*`name`*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`name`	*String, optional*	*The name of the layer.*

ui.Map.Layer.setOpacity

Sets the opacity of the layer.

Returns this map layer.

Usage	Returns
`Layer.setOpacity(`*`opacity`*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`opacity`	*Number, optional*	*The layer's opacity represented as a number between 0 and 1.*

ui.Map.Layer.setShown

Sets the visibility of the layer.

Returns this map layer.

Usage	Returns
`Layer.setShown(`*`shown`*`)`	ui.Map.AbstractLayer

Argument	Type	Details
this: `ui.map.abstractlayer`	ui.Map.AbstractLayer	The ui.Map.AbstractLayer instance.
`shown`	*Boolean, optional*	*Whether the layer is shown.*

ui.Map.Layer.setVisParams

Sets the layer's visualization parameters.

Returns this map layer.

Usage	Returns
`Layer.setVisParams(`*`visParams`*`)`	ui.Map.Layer

Argument	Type	Details
this: `ui.map.layer`	ui.Map.Layer	The ui.Map.Layer instance.

Argument	Type	Details
`visParams`	*FeatureVisualizationParameters|ImageVisualizationParameters, optional*	*The visualization parameters. See ee.data.getMapId() docs.*

ui.Map.Linker

A utility for creating linked maps.

Usage	Returns
`ui.Map.Linker(`*`maps`*`, `*`event`*`)`	ui.Map.Linker

Argument	Type	Details
`maps`	*List, optional*	*A list of maps to link.*
`event`	*String, optional*	*The event to link across the maps. Defaults to "change-bounds". Possible events comprise:* • *change-bounds* • *change-center* • *change-zoom*

ui.Map.Linker.add

Appends an element to the list.

Returns this ui.data.ActiveList.

Usage	Returns
`Linker.add(el)`	ui.data.ActiveList

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`el`	Object	The element to add.

ui.Map.Linker.forEach

Iterates over each element, calling the provided callback. The callback is called for each element like: callback(element, index).

Usage	Returns
`Linker.forEach(callback)`	

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`callback`	Function	

ui.Map.Linker.get

Returns the element at the specified index.

Usage	Returns
`Linker.get(index)`	Object

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`index`	Number	The index of the element to return.

ui.Map.Linker.getJsArray

Returns the list as a JS array.

Usage	Returns
`Linker.getJsArray()`	List

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.

ui.Map.Linker.insert

Inserts an element at the specified index and shifts the rest of the list. If the specified index is greater than the length of the list, the element will be appended to the list.

Returns this ui.data.ActiveList.

Usage	Returns
`Linker.insert(index, el)`	ui.data.ActiveList

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`index`	Number	The index at which to insert the element.
`el`	Object	The element to insert.

ui.Map.Linker.length

Returns the number of elements in the list.

Usage	Returns
`Linker.length()`	Number

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.

ui.Map.Linker.remove

Removes the specified element from the list.

Returns the removed element or null if the element was not present in the list.

Usage	Returns
`Linker.remove(el)`	Object

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`el`	Object	The element to remove.

ui.Map.Linker.reset

Replaces all elements in list with a new list or, if no list is provided, removes all elements from list.

Returns the elements in the list after the reset is applied.

Usage	Returns
`Linker.reset(`*`list`*`)`	List

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`list`	*List, optional*	*A list of elements.*

ui.Map.Linker.set

Sets an element at the specified index. If the index exceeds that of the list's last element, the element will be added to the end of the list.

Returns this ui.data.ActiveList.

Usage	Returns
`Linker.set(index, el)`	ui.data.ActiveList

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`index`	Number	The index to overwrite.
`el`	Object	The element to set.

ui.Map.add

Adds an item to the map. Can also be used to add widgets like ui.Label as well as some non-widget objects like ui.Map.Layer.

Returns the map.

Usage	Returns
`Map.add(item)`	ui.Map

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`item`	Object	The item to add.

ui.Map.addLayer

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

Returns the new map layer.

Usage	Returns
`Map.addLayer(eeObject, visParams, name, shown, opacity)`	ui.Map.Layer

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`eeObject`	Collection|Feature|Image|MapId	The object to add to the map.
`visParams`	*FeatureVisualizationParameters|ImageVisualizationParameters, optional*	*The visualization parameters. For Images and ImageCollection, see ee.data.getMapId for valid*

Argument	Type	Details
		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".*
`shown`	*Boolean, optional*	*A flag indicating whether the layer should be on by default.*
`opacity`	*Number, optional*	*The layer's opacity represented as a number between 0 and 1. Defaults to 1.*

ui.Map.centerObject

Centers the map view on a given object.

Caution: providing a large or complex collection as input can result in poor performance. Collating the geometry of collections does not scale well; use the smallest collection (or geometry) that is required to achieve the desired outcome.

Returns this ui.Map.

Usage	Returns
`Map.centerObject(object, zoom, onComplete)`	ui.Map

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`object`	Element|Geometry	An object to center on - a geometry, image or feature.
`zoom`	*Number, optional*	*The zoom level, from 0 to 24. If unspecified, computed based on the object's bounding box.*
`onComplete`	*Function, optional*	*A callback which is triggered after the recentering completes successfully. Passing this parameter causes the `centerObject` operation to run asynchronously.*

ui.Map.clear

Clears the map by removing all layers, listeners, and widgets and restoring the options to their defaults.

Returns the map.

Usage	Returns
`Map.clear()`	ui.Map

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.

ui.Map.drawingTools

Returns the map's drawing tools, which can be used to create and edit shapes on the map. Adds the drawing tools to the map if none exist.

Usage	Returns
`Map.drawingTools()`	ui.Map.DrawingTools

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.

ui.Map.getBounds

Returns the bounds of the current map view, as a list in the format [west, south, east, north] in degrees.

Usage	Returns
`Map.getBounds(`*`asGeoJSON`*`)`	GeoJSONGeometry|List|String

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`asGeoJSON`	*Boolean, optional*	*If true, returns map bounds as GeoJSON.*

ui.Map.getCenter

Returns the coordinates at the center of the map.

Usage	Returns
`Map.getCenter()`	Geometry.Point

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.

ui.Map.getScale

Returns the approximate pixel scale of the current map view, in meters.

Usage	Returns
`Map.getScale()`	Number|String

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.

ui.Map.getZoom

Returns the current zoom level of the map.

Usage	Returns
`Map.getZoom()`	Number

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.

ui.Map.insert

Inserts a widget into to the panel at the specified index.

Returns this panel.

Usage	Returns
`Map.insert(index, widget)`	ui.Panel

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.
`index`	Number	The index at which to insert the widget.
`widget`	ui.Widget	The widget to insert.

ui.Map.layers

Returns the list of layers associated with the map.

Usage	Returns
`Map.layers()`	ui.data.ActiveList

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.

ui.Map.onChangeBounds

Registers a callback that's fired when the map bounds change. This is fired during pan, zoom, and when the map's bounds are changed programmatically.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onChangeBounds(callback)`	String

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`callback`	Function	The callback to fire when the map bounds change. The callback is passed two parameters: an object containing the coordinates of the new map center (with keys lon, lat, and zoom) and the map widget itself.

ui.Map.onChangeCenter

Registers a callback that's fired when the map center changes. This is fired during pan or when the map's center is changed programmatically.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onChangeCenter(callback)`	String

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`callback`	Function	The callback to fire when the map center changes. The callback is passed two parameters: an object containing the coordinates of the new center (with keys lon and lat) and the map widget itself.

ui.Map.onChangeZoom

Registers a callback that's fired when the map zoom level changes.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onChangeZoom(callback)`	String

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`callback`	Function	The callback to fire when the map zoom change. The callback is passed two parameters: the new zoom level and the map widget itself.

ui.Map.onClick

Registers a callback that's fired when the map is clicked.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onClick(callback)`	String

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`callback`	Function	The callback to fire when the map is clicked. The callback is passed an object containing the coordinates of the clicked point on the map (with keys lon and lat) and the map widget itself.

ui.Map.onIdle

Registers a callback that's fired when the map stops moving.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onIdle(callback)`	String

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`callback`	Function	The callback to fire when the map becomes idle. The callback is passed two parameters: an object containing the coordinates of the map center (with keys lon, lat, and zoom) and the map widget itself.

ui.Map.onTileLoaded

Registers a callback that's fired when a map tile has been loaded.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Map.onTileLoaded(callback)`	String

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`callback`	Function	Called with an array of per layer values. Each value is the fraction of tiles still pending: a value of 0 means there are no more tiles to load for the layer.

ui.Map.remove

Removes the given item from the map, if it exists.

Returns the removed item or null if it hadn't been added to the map.

Usage	Returns
`Map.remove(item)`	Object

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`item`	Object	The item to remove.

ui.Map.setCenter

Centers the map view at the given coordinates with the given zoom level. If no zoom level is provided, it uses the most recent zoom level on the map.

Returns this ui.Map.

Usage	Returns
`Map.setCenter(lon, lat, `*`zoom`*`)`	ui.Map

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`lon`	Number	The longitude of the center, in degrees.
`lat`	Number	The latitude of the center, in degrees.
`zoom`	*Number, optional*	*The zoom level, from 0 to 24.*

ui.Map.setControlVisibility

Sets the visibility of the controls on the map.

Returns this ui.Map.

Usage	Returns
`Map.setControlVisibility(`*`all`*`, `*`layerList`*`, `*`zoomControl`*`, `*`scaleControl`*`, `*`mapTypeControl`*`, `*`fullscreenControl`*`, `*`drawingToolsControl`*`)`	ui.Map

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`all`	*Boolean, optional*	*Whether to show all controls. False hides all controls; true shows all controls. Overridden by individually set parameters. Note that setting this explicitly will affect any additional controls added in the future.*
`layerList`	*Boolean, optional*	*When false, hides the layer list panel or, when true, allows the layer list panel's visibility to be determined by the presence of layers in the list. The default is to show the list.*
`zoomControl`	*Boolean, optional*	*Whether the zoom control is visible. Defaults to true.*
`scaleControl`	*Boolean, optional*	*Whether to show the control which indicates the scale at the map's current zoom level. Defaults to true.*
`mapTypeControl`	*Boolean, optional*	*Whether to show the control that allows the user to change the base map. Defaults to true.*
`fullscreenControl`	*Boolean, optional*	*Whether to show the control that allows the user to make the map full-screen. Defaults to true.*
`drawingToolsControl`	*Boolean, optional*	*Whether to show the control that allows the user to add or edit the geometry drawing tools. Defaults to true if the drawing tools were previously added to the map. Ignored if the drawing tools were not previously added to the map.*

ui.Map.setGestureHandling

Controls how gestures are handled on the map.

See https://developers.google.com/maps/documentation/javascript/reference/map#MapOptions.gestureHandling.

Usage	Returns
`Map.setGestureHandling(option)`	

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`option`	String	The option that controls how gestures are handled on the map. Allowed values: • "cooperative": Scroll events and one-finger touch gestures scroll the page, and do not zoom or pan the map. Two-finger touch gestures pan and zoom the map. Scroll events with a ctrl key or ⌘ key pressed zoom the map. In this mode the map cooperates with the page. • "greedy": All touch gestures and scroll events pan or zoom the map. • "none": The map cannot be panned or zoomed by user gestures. • "auto": (default) Gesture handling is either cooperative or greedy, depending on whether the page is scrollable or in an iframe.

ui.Map.setLocked

Limits panning and zooming on the map.

- To lock both panning and zooming, set locked to true and nothing else.

- To allow panning and limit the min and max zoom, set locked to false and supply the minZoom and maxZoom parameters.

- To disallow panning and limit min and max zoom, set locked to true and supply the minZoom and maxZoom parameters.

- To reset the map to default, set locked to false and nothing else.

Usage	Returns
`Map.setLocked(locked, `*`minZoom`*`, `*`maxZoom`*`)`	

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`locked`	Boolean	Whether the map should be locked or not.
`minZoom`	*Number, optional*	*(optional) The minimum zoom for the map, between 0 and 24, inclusive.*
`maxZoom`	*Number, optional*	*(optional) The maximum zoom for the map, between 0 and 24, inclusive.*

ui.Map.setOptions

Modifies the Google Maps basemap. Allows for: 1) Setting the current MapType. 2) Providing custom styles for the basemap (MapTypeStyles). 3) Setting the list of available mapTypesIds for the basemap.

If called with no parameters, resets the map type to the Google Maps default.

Returns this ui.Map.

Usage	Returns
`Map.setOptions(`*`mapTypeId`*`, `*`styles`*`, `*`types`*`)`	ui.Map

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`mapTypeId`	*String, optional*	*A mapTypeId to set the basemap to. Can be one of "ROADMAP", "SATELLITE", "HYBRID", or "TERRAIN" to select one of the standard Google Maps API map types, or one of the keys specified in the opt_styles dictionary. If left as null and only 1 style is specified in opt_styles, that style will be used.*
`styles`	*Object, optional*	*A dictionary of custom MapTypeStyle objects keyed with a name that will appear in the map's Map Type Controls. See: https://developers.google.com/maps/documentation/javascript/reference#MapTypeStyle*
`types`	*List, optional*	*A list of mapTypeIds to make available. If omitted, but opt_styles is specified, appends all of the style keys to the standard Google Maps API map types.*

ui.Map.setZoom

Sets the zoom level of the map.

Returns this ui.Map.

Usage	Returns
`Map.setZoom(zoom)`	ui.Map

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`zoom`	Number	The zoom level, from 0 to 24, to set for the map.

ui.Map.style

Returns the map's style ActiveDictionary, which can be modified to update the map's styles.

In addition to the standard UI API styles listed in the ui.Panel.style() documentation, ui.Map supports the following custom style option:

- cursor, which can be 'crosshair' or 'hand' (default)

Usage	Returns
`Map.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.

ui.Map.unlisten

Deletes callbacks.

Usage	Returns
`Map.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.map`	ui.Map	The ui.Map instance.
`idOrType`	*String, optional*	*Either an ID returned by listen() when a callback was registered, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks registered with that event type are deleted. If nothing is passed, all callbacks are deleted.*

ui.Map.widgets

Returns the list of widgets currently in the panel.

Usage	Returns
`Map.widgets()`	ui.data.ActiveList

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.

ui.Panel

A widget that can hold other widgets. Use panels to construct complex combinations of nested widgets.

Panels can be added to ui.root but not printed to the console with print().

Usage	Returns
`ui.Panel(`*`widgets`*`, `*`layout`*`, `*`style`*`)`	ui.Panel

Argument	Type	Details
`widgets`	*List, optional*	*The list of widgets or a single widget to add to the panel. Defaults to an empty array.*
`layout`	*String|ui.Panel.Layout, optional*	*The layout to use for this panel. If a string is passed in, it's taken as a shortcut to the layout constructor with that name. Defaults to 'flow'.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this widget. See style() documentation.*

ui.Panel.Layout.absolute

Returns a layout that places its widgets absolutely relative to the panel.

An added widget's "position" style property determines how it is placed. The following positions are supported:

- top-left, top-center, top-right

- middle-left, middle-right

- bottom-left, bottom-center, bottom-right

If no position is specified, the widget will be placed behind (that is, with a lower z-index than) the positioned widgets.

Usage	Returns
`ui.Panel.Layout.absolute()`	ui.Panel.Layout

No arguments.

ui.Panel.Layout.flow

Returns a layout that places its widgets in a flow, either horizontal or vertical. By default, widgets take up their natural space within a flow layout panel. Set the "stretch" style property on an added widget to stretch it to fill available space in the relevant direction: - horizontal, vertical, both When multiple widgets are stretched, the available space is split equally among them. Panels are widgets themselves and can be stretched by specifying a "stretch" style property.

Usage	Returns
`ui.Panel.Layout.flow(`*`direction`*`, `*`wrap`*`)`	ui.Panel.Layout

Argument	Type	Details
`direction`	*String, optional*	*The direction of the flow. One of 'horizontal' or 'vertical'. Defaults to 'vertical'.*
`wrap`	*Boolean, optional*	*Whether to wrap children in the layout if there are too many to show in one line. Defaults to false.*

ui.Panel.add

Adds a widget to the panel.

Returns this panel.

Usage	Returns
`Panel.add(widget)`	ui.Panel

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.
`widget`	ui.Widget	The widget to be added.

ui.Panel.clear

Removes all widgets from the panel.

Returns this panel.

Usage	Returns
`Panel.clear()`	ui.Panel

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.

ui.Panel.getLayout

Gets the panel's layout.

Usage	Returns
`Panel.getLayout()`	ui.Panel.Layout

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.

ui.Panel.insert

Inserts a widget into to the panel at the specified index.

Returns this panel.

Usage	Returns
`Panel.insert(index, widget)`	ui.Panel

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.
`index`	Number	The index at which to insert the widget.
`widget`	ui.Widget	The widget to insert.

ui.Panel.remove

Removes the given widget from the panel, if it exists.

Returns whether the widget was successfully removed.

Usage	Returns
`Panel.remove(widget)`	Boolean

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.
`widget`	ui.Widget	The widget to remove.

ui.Panel.setLayout

Sets the panel's layout.

Returns this panel.

Usage	Returns
`Panel.setLayout(layout)`	ui.Panel

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.
`layout`	ui.Panel.Layout	The new layout.

ui.Panel.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')

- width, maxWidth, minWidth (e.g. '100px')

- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')

- color, backgroundColor (e.g. 'red' or '#FF0000')

- border (e.g. '1px solid black')

- fontSize (e.g. '24px')

- fontStyle (e.g. 'italic')

- fontWeight (e.g. 'bold' or '100')

- fontFamily (e.g. 'monospace' or 'serif')

- textAlign (e.g. 'left' or 'center')

- textDecoration (e.g. 'underline' or 'line-through')

- whiteSpace (e.g. 'nowrap' or 'pre')

- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')

- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Panel.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Panel.widgets

Returns the list of widgets currently in the panel.

Usage	Returns
`Panel.widgets()`	ui.data.ActiveList

Argument	Type	Details
this: `ui.panel`	ui.Panel	The ui.Panel instance.

ui.Select

A printable select menu with a callback.

Usage	Returns
`ui.Select(`*`items`*`, `*`placeholder`*`, `*`value`*`, `*`onChange`*`, `*`disabled`*`, `*`style`*`)`	ui.Select

Argument	Type	Details
`items`	*List, optional*	*The list of options to add to the select. Defaults to an empty array.*
`placeholder`	*String, optional*	*The placeholder shown when no value is selected. Defaults to "Select a value...".*
`value`	*String, optional*	*The select's value. Defaults to null.*
`onChange`	*Function, optional*	*The callback to fire when an item is selected. The callback is passed the currently selected value and the select widget.*
`disabled`	*Boolean, optional*	*Whether the select is disabled. Defaults to false.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this widget. See style() documentation.*

ui.Select.getDisabled

Returns whether the select is disabled.

Usage	Returns
`Select.getDisabled()`	Boolean

Argument	Type	Details
this: `ui.select`	ui.Select	The ui.Select instance.

ui.Select.getPlaceholder

Returns the select's placeholder text.

Usage	Returns
`Select.getPlaceholder()`	String

Argument	Type	Details
this: `ui.select`	ui.Select	The ui.Select instance.

ui.Select.getValue

Returns the currently selected value.

Usage	Returns
`Select.getValue()`	String

Argument	Type	Details
this: `ui.select`	ui.Select	The ui.Select instance.

ui.Select.items

See ui.data.ActiveList.

Returns the list of items in the selection menu.

Usage	Returns
`Select.items()`	ui.data.ActiveList

Argument	Type	Details
this: `ui.select`	ui.Select	The ui.Select instance.

ui.Select.onChange

Registers a callback that's fired when an item is selected.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Select.onChange(callback)`	String

Argument	Type	Details
this: `ui.select`	ui.Select	The ui.Select instance.
`callback`	Function	The callback to fire when an item is selected. The callback is passed the currently selected value and the select widget.

ui.Select.setDisabled

Sets whether the select is disabled.

Returns this select.

Usage	Returns
`Select.setDisabled(disabled)`	ui.Select

Argument	Type	Details
this: `ui.select`	ui.Select	The ui.Select instance.
`disabled`	Boolean	Whether the select is disabled.

ui.Select.setPlaceholder

Sets the select's placeholder text, which is shown when no value is selected.

Returns this select.

Usage	Returns
`Select.setPlaceholder(placeholder)`	ui.Select

Argument	Type	Details
this: `ui.select`	ui.Select	The ui.Select instance.

Argument	Type	Details
`placeholder`	String	The select's placeholder text.

ui.Select.setValue

Sets the selected value.

Returns this select.

Usage	Returns
`Select.setValue(value, `*`trigger`*`)`	ui.Select

Argument	Type	Details
this: `ui.select`	ui.Select	The ui.Select instance.
`value`	String	The value to select.
`trigger`	*Boolean, optional*	*Whether to trigger onChange callbacks when the value property changes. Defaults to true.*

ui.Select.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')

- width, maxWidth, minWidth (e.g. '100px')

- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')

- color, backgroundColor (e.g. 'red' or '#FF0000')

- border (e.g. '1px solid black')

- fontSize (e.g. '24px')

- fontStyle (e.g. 'italic')

- fontWeight (e.g. 'bold' or '100')

- fontFamily (e.g. 'monospace' or 'serif')

- textAlign (e.g. 'left' or 'center')

- textDecoration (e.g. 'underline' or 'line-through')

- whiteSpace (e.g. 'nowrap' or 'pre')

- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')

- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Select.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Select.unlisten

Deletes callbacks.

Usage	Returns
`Select.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.Slider

A draggable target that ranges linearly between two numeric values. The value of the slider is displayed as a label alongside it.

Usage	Returns
`ui.Slider(`*`min`*`, `*`max`*`, `*`value`*`, `*`step`*`, `*`onChange`*`, `*`direction`*`, `*`disabled`*`, `*`style`*`)`	ui.Slider

Argument	Type	Details
`min`	*Number, optional*	*The minimum value. Defaults to 0.*
`max`	*Number, optional*	*The maximum value. Defaults to 1.*
`value`	*Number, optional*	*The initial value. Defaults to 0.*
`step`	*Number, optional*	*The step size for the slider. Defaults to 0.01.*
`onChange`	*Function, optional*	*A callback to fire when the slider's state changes. The callback is passed the slider's current value and the slider widget.*
`direction`	*String, optional*	*The direction of the slider. One of 'horizontal' or 'vertical'. Defaults to 'horizontal'.*
`disabled`	*Boolean, optional*	*Whether the slider is disabled. Defaults to false.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this widget. See style() documentation.*

ui.Slider.getDisabled

Returns whether the slider is disabled.

Usage	Returns
`Slider.getDisabled()`	Boolean

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.

ui.Slider.getMax

Returns the slider's maximum value.

Usage	Returns
`Slider.getMax()`	Number

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.

ui.Slider.getMin

Returns the slider's minimum value.

Usage	Returns
`Slider.getMin()`	Number

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.

ui.Slider.getStep

Returns the slider's step value.

Usage	Returns
`Slider.getStep()`	Number

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.

ui.Slider.getValue

Returns the current slider value.

Usage	Returns
`Slider.getValue()`	Number

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.

ui.Slider.onChange

Registers a callback that's fired when the slider's state changes. If the change is due to the user dragging the slider, the event will not fire until the drag completes.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Slider.onChange(callback)`	String

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.
`callback`	Function	The callback to fire when the slider's state changes. The callback is passed the slider's current value and the slider widget.

ui.Slider.onSlide

Registers a callback that's fired when the slider's state changes. The callback will be invoked repeatedly while the user is dragging the slider.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Slider.onSlide(callback)`	String

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.

Argument	Type	Details
`callback`	Function	The callback to fire when the slider's state changes. The callback is passed the slider's current value.

ui.Slider.setDisabled

Sets whether the slider is disabled.

Returns this slider.

Usage	Returns
`Slider.setDisabled(disabled)`	ui.Slider

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.
`disabled`	Boolean	Whether the slider is disabled.

ui.Slider.setMax

Sets the maximum value of the slider.

Returns this slider.

Usage	Returns
`Slider.setMax(value)`	ui.Slider

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.
`value`	Number	The slider's maximum value.

ui.Slider.setMin

Sets the minimum value of the slider.

Returns this slider.

Usage	Returns
`Slider.setMin(value)`	ui.Slider

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.
`value`	Number	The slider's minimum value.

ui.Slider.setStep

Sets the step value of the slider.

Returns this slider.

Usage	Returns
`Slider.setStep(value)`	ui.Slider

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.
`value`	Number	The slider's step value.

ui.Slider.setValue

Set the value of the slider.

Returns this slider.

Usage	Returns
`Slider.setValue(value, `*`trigger`*`)`	ui.Slider

Argument	Type	Details
this: `ui.slider`	ui.Slider	The ui.Slider instance.
`value`	Number	The value to slider.
`trigger`	*Boolean, optional*	*Whether to trigger onChange callbacks when the value property changes. Defaults to true.*

ui.Slider.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')
- width, maxWidth, minWidth (e.g. '100px')
- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')
- color, backgroundColor (e.g. 'red' or '#FF0000')
- border (e.g. '1px solid black')
- fontSize (e.g. '24px')
- fontStyle (e.g. 'italic')
- fontWeight (e.g. 'bold' or '100')
- fontFamily (e.g. 'monospace' or 'serif')
- textAlign (e.g. 'left' or 'center')
- textDecoration (e.g. 'underline' or 'line-through')
- whiteSpace (e.g. 'nowrap' or 'pre')
- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')
- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Slider.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Slider.unlisten

Deletes callbacks.

Usage	Returns
`Slider.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.SplitPanel

A widget containing two panels with a divider between them. The divider can be dragged, allowing the panels to be resized. One or both panels may be ui.Map objects.

By default the layout initializes with a 50/50 split. The width and max/minWidth styles on the panels control the split sizing for horizontal orientations. Similarly, use height and max/minHeight for vertical. These can be given in pixels as '{n}px' or as a percentage of the containing SplitPanel as '{n}%'.

Note that the given size for the second panel will be ignored if the first panel size is specified, since the overall width of the split panel is controlled independently. Max/min sizes may be set for both panels.

Usage	Returns
`ui.SplitPanel(`*`firstPanel`*`, `*`secondPanel`*`, `*`orientation`*`, `*`wipe`*`, `*`style`*`)`	ui.SplitPanel

Argument	Type	Details
`firstPanel`	*ui.Panel, optional*	*The left or top panel. Defaults to a new instance of ui.Panel.*
`secondPanel`	*ui.Panel, optional*	*The bottom or right panel. Defaults to a new instance of ui.Panel.*
`orientation`	*String, optional*	*One of "horizontal" or "vertical". Defaults to "horizontal".*
`wipe`	*Boolean, optional*	*Whether to enable the wiping effect. When this mode is enabled, both panels take up all available space, and dragging the divider doesn't set the size of the panels but rather determines how much of each panel is shown. This effect is analogous to a "wipe transition". This mode is useful for comparing two maps. Defaults to false.*

Argument	Type	Details
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this panel. Defaults to an empty object.*

ui.SplitPanel.getFirstPanel

Returns the first panel in the split panel.

Usage	Returns
`SplitPanel.getFirstPanel()`	ui.Panel

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.

ui.SplitPanel.getOrientation

Returns the panel's orientation.

Usage	Returns
`SplitPanel.getOrientation()`	String

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.

ui.SplitPanel.getPanel

Returns the requested panel in the split panel.

Usage	Returns
`SplitPanel.getPanel(index)`	ui.Panel

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.
`index`	Number	0 for top or left panel, 1 for bottom or right panel.

ui.SplitPanel.getSecondPanel

Returns the second panel in the split panel.

Usage	Returns
`SplitPanel.getSecondPanel()`	ui.Panel

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.

ui.SplitPanel.getWipe

Returns whether the wiping effect is enabled.

Usage	Returns
`SplitPanel.getWipe()`	Boolean

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.

ui.SplitPanel.setFirstPanel

Returns this split panel.

Usage	Returns
`SplitPanel.setFirstPanel(panel)`	ui.SplitPanel

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.
`panel`	ui.Panel	The panel to display left or on top of the split.

ui.SplitPanel.setOrientation

Sets the panel's orientation; one of "horizontal" or "vertical".

Returns this split panel.

Usage	Returns
`SplitPanel.setOrientation(orientation)`	ui.SplitPanel

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.
`orientation`	String	The new orientation.

ui.SplitPanel.setPanel

Returns the requested panel in the split panel.

Usage	Returns
`SplitPanel.setPanel(index, panel)`	ui.Panel

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.
`index`	Number	0 for top or left panel, 1 for bottom or right panel.
`panel`	ui.Panel	The panel to add to the split panel.

ui.SplitPanel.setSecondPanel

Returns this split panel.

Usage	Returns
`SplitPanel.setSecondPanel(panel)`	ui.SplitPanel

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.
`panel`	ui.Panel	The panel to display right of or below the split.

ui.SplitPanel.setWipe

Enables or disables the wiping effect.

Returns this split panel.

Usage	Returns
`SplitPanel.setWipe(wipe)`	ui.SplitPanel

Argument	Type	Details
this: `ui.splitpanel`	ui.SplitPanel	The ui.SplitPanel instance.
`wipe`	Boolean	Whether to enable the wiping effect.

ui.SplitPanel.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')

- width, maxWidth, minWidth (e.g. '100px')

- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')

- color, backgroundColor (e.g. 'red' or '#FF0000')

- border (e.g. '1px solid black')

- fontSize (e.g. '24px')

- fontStyle (e.g. 'italic')

- fontWeight (e.g. 'bold' or '100')

- fontFamily (e.g. 'monospace' or 'serif')

- textAlign (e.g. 'left' or 'center')

- textDecoration (e.g. 'underline' or 'line-through')

- whiteSpace (e.g. 'nowrap' or 'pre')

- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')

- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`SplitPanel.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.SplitPanel.unlisten

Deletes callbacks.

Usage	Returns
`SplitPanel.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.Textbox

A textbox that enables the user to input text information.

Usage	Returns
`ui.Textbox(`*`placeholder`*`, `*`value`*`, `*`onChange`*`, `*`disabled`*`, `*`style`*`)`	ui.Textbox

Argument	Type	Details
`placeholder`	*String, optional*	*The placeholder text to display when the textbox is empty. Defaults to none.*
`value`	*String, optional*	*The textbox's value. Defaults to none.*
`onChange`	*Function, optional*	*The callback to fire when the text changes. The callback is passed the text currently in the textbox and the textbox widget.*
`disabled`	*Boolean, optional*	*Whether the textbox is disabled. Defaults to false.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this widget. See style() documentation.*

ui.Textbox.getDisabled

Returns whether the textbox is disabled.

Usage	Returns
`Textbox.getDisabled()`	Boolean

Argument	Type	Details
this: `ui.textbox`	ui.Textbox	The ui.Textbox instance.

ui.Textbox.getPlaceholder

Returns the textbox's placeholder text.

Usage	Returns
`Textbox.getPlaceholder()`	String

Argument	Type	Details
this: `ui.textbox`	ui.Textbox	The ui.Textbox instance.

ui.Textbox.getValue

Returns the value of the textbox.

Usage	Returns
`Textbox.getValue()`	String

Argument	Type	Details
this: `ui.textbox`	ui.Textbox	The ui.Textbox instance.

ui.Textbox.onChange

Registers a callback that's called when text in the textbox changes.

In particular, the callback is called when:

- The user types a new value and then either the textbox loses focus or the user presses enter.

- A new value is set programmatically with set('value', newValue).

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Textbox.onChange(callback)`	String

Argument	Type	Details
this: `ui.textbox`	ui.Textbox	The ui.Textbox instance.
`callback`	Function	The callback to fire when the text changes. The callback is passed the text currently in the textbox and the textbox widget.

ui.Textbox.setDisabled

Sets whether the textbox is disabled.

Returns this textbox.

Usage	Returns
`Textbox.setDisabled(disabled)`	ui.Textbox

Argument	Type	Details
this: `ui.textbox`	ui.Textbox	The ui.Textbox instance.
`disabled`	Boolean	Whether the textbox is disabled.

ui.Textbox.setPlaceholder

Sets the textbox's placeholder text, which is shown when no text is entered.

Returns this select.

Usage	Returns
`Textbox.setPlaceholder(placeholder)`	ui.Textbox

Argument	Type	Details
this: `ui.textbox`	ui.Textbox	The ui.Textbox instance.
`placeholder`	String	The select's placeholder text.

ui.Textbox.setValue

Sets the value of the textbox.

Returns this textbox.

Usage	Returns
`Textbox.setValue(value, `*`trigger`*`)`	ui.Textbox

Argument	Type	Details
this: `ui.textbox`	ui.Textbox	The ui.Textbox instance.
`value`	String	The value of the textbox.
`trigger`	*Boolean, optional Whether to trigger onChange callbacks when the value property changes. Defaults to true.*	

ui.Textbox.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')
- width, maxWidth, minWidth (e.g. '100px')
- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')
- color, backgroundColor (e.g. 'red' or '#FF0000')
- border (e.g. '1px solid black')
- fontSize (e.g. '24px')
- fontStyle (e.g. 'italic')
- fontWeight (e.g. 'bold' or '100')
- fontFamily (e.g. 'monospace' or 'serif')
- textAlign (e.g. 'left' or 'center')
- textDecoration (e.g. 'underline' or 'line-through')
- whiteSpace (e.g. 'nowrap' or 'pre')
- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')
- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Textbox.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Textbox.unlisten

Deletes callbacks.

Usage	Returns
`Textbox.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.Thumbnail

A fixed-size thumbnail image generated asynchronously from an ee.Image.

Usage	Returns
`ui.Thumbnail(`*`image`*`, `*`params`*`, `*`onClick`*`, `*`style`*`)`	ui.Thumbnail

Argument	Type	Details
`image`	*Image, optional*	*The ee.Image from which to generate the thumbnail. Defaults to an empty ee.Image.*
`params`	*Object, optional*	*For an explanation of the possible parameters, see ui.Thumbnail.setParams(). Defaults to an empty object.*
`onClick`	*Function, optional*	*A callback fired when the thumbnail is clicked.*
`style`	*Object, optional*	*An object of allowed CSS styles with their values to be set for this label. Defaults to an empty object.*

ui.Thumbnail.getImage

Returns the ee.Image for the thumbnail.

Usage	Returns
`Thumbnail.getImage()`	Image|ImageCollection

Argument	Type	Details
this: `ui.thumbnail`	ui.Thumbnail	The ui.Thumbnail instance.

ui.Thumbnail.getParams

See ee.Image.prototype.getThumbnailURL.

Returns the parameters used in generating the thumbnail.

Usage	Returns
`Thumbnail.getParams()`	Object

Argument	Type	Details
this: `ui.thumbnail`	ui.Thumbnail	The ui.Thumbnail instance.

ui.Thumbnail.onClick

Registers a callback that's fired when the thumbnail is clicked.

Returns an ID which can be passed to unlisten() to unregister the callback.

Usage	Returns
`Thumbnail.onClick(callback)`	String

Argument	Type	Details
this: `ui.thumbnail`	ui.Thumbnail	The ui.Thumbnail instance.
`callback`	Function	The callback to fire when the thumbnail is clicked. The callback is passed the thumbnail widget.

ui.Thumbnail.setImage

Sets the ee.Image used to generate the thumbnail.

Returns this thumbnail.

Usage	Returns
`Thumbnail.setImage(image)`	ui.Thumbnail

Argument	Type	Details
this: `ui.thumbnail`	ui.Thumbnail	The ui.Thumbnail instance.
`image`	Image	The image from which to generate the thumbnail.

ui.Thumbnail.setParams

Sets the parameters used to generate the thumbnail.

Returns this thumbnail.

Usage	Returns
`Thumbnail.setParams(params)`	ui.Thumbnail

Argument	Type	Details
this: `ui.thumbnail`	ui.Thumbnail	The ui.Thumbnail instance.
`params`	Object	The parameters used in generating the thumbnail. `dimensions` (a number or pair of numbers in format WIDTHxHEIGHT) Maximum dimensions of the thumbnail to render, in pixels. If only one number is passed, it is used as the maximum, and the other dimension is computed by proportional scaling. `region` (E,S,W,N or GeoJSON) Geospatial region of the image to render. By default, the whole image. `format` (string) Either 'png' or 'jpg'. `bands` (comma-separated strings) Comma-delimited list of band names to be mapped to RGB. `min` (comma-separated numbers) Value (or one per band) to map onto 00. `max` (comma-separated numbers) Value (or one per band) to map onto FF.

Argument	Type	Details
		`gain` (comma-separated numbers) Gain (or one per band) to map onto 00-FF.
		`bias` (comma-separated numbers) Offset (or one per band) to map onto 00-FF.
		`gamma` (comma-separated numbers) Gamma correction factor (or one per band)
		`palette` (comma-separated strings) List of CSS-style color strings (single-band previews only).
		`opacity` (number) a number between 0 and 1 for opacity.
		`version` (number) Version number of image (or latest).

ui.Thumbnail.style

Returns the widget's style ActiveDictionary, which can be modified to update the widget's styles.

Properties which behave like their CSS counterparts:

- height, maxHeight, minHeight (e.g. '100px')

- width, maxWidth, minWidth (e.g. '100px')

- padding, margin (e.g. '4px 4px 4px 4px' or simply '4px')

- color, backgroundColor (e.g. 'red' or '#FF0000')

- border (e.g. '1px solid black')

- fontSize (e.g. '24px')

- fontStyle (e.g. 'italic')

- fontWeight (e.g. 'bold' or '100')

- fontFamily (e.g. 'monospace' or 'serif')

- textAlign (e.g. 'left' or 'center')

- textDecoration (e.g. 'underline' or 'line-through')

- whiteSpace (e.g. 'nowrap' or 'pre')

- shown (true or false)

Supported custom layout properties (see ui.Panel.Layout documentation):

- stretch ('horizontal', 'vertical', 'both')

- position ('top-right', 'top-center', 'top-left', 'bottom-right', ...)

Usage	Returns
`Thumbnail.style()`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.

ui.Thumbnail.unlisten

Deletes callbacks.

Usage	Returns
`Thumbnail.unlisten(`*`idOrType`*`)`	

Argument	Type	Details
this: `ui.widget`	ui.Widget	The ui.Widget instance.
`idOrType`	*String, optional*	*Either an ID returned by an onEventType() function during callback registration, an event type, or nothing. If an ID is passed, the corresponding callback is deleted. If an event type is passed, all callbacks for that type are deleted. If nothing is passed, all callbacks are deleted.*

ui.data.ActiveDictionary

A dictionary-like container for data for use in UI components.

When a property of a ui.data.ActiveDictionary (e.g. myButton.style()) is updated, the component it belongs to is automatically updated. For example, myButton.style().set('color', 'red') would change the color of button's text to red.

Usage	Returns
`ui.data.ActiveDictionary(`*`object`*`, `*`allowedProperties`*`)`	ui.data.ActiveDictionary

Argument	Type	Details
`object`	*Object, optional A JavaScript object with properties and values to initialize this object with.*	

Argument	Type	Details
`allowedProperties`	*List, optional*	*An array of allowed properties for this object. If undefined, then any property is allowed.*

ui.data.ActiveDictionary.get

Returns either a clone of this object or, if a key is provided, the value of the property with the passed-in key. Look at the constructor's parameters to see which properties are available.

Usage	Returns
`ActiveDictionary.get(`*`key`*`)`	Object

Argument	Type	Details
this: `ui.data.activedictionary`	ui.data.ActiveDictionary	The ui.data.ActiveDictionary instance.
`key`	*String, optional*	*The key of the property to retrieve.*

ui.data.ActiveDictionary.set

Sets the value of a given property. Throws an error if the key provided is not supported by the object. Look at the constructor's parameters to see which properties can be set.

Returns this ui.data.ActiveDictionary.

Usage	Returns
`ActiveDictionary.set(keyOrDict, `*`value`*`)`	ui.data.ActiveDictionary

Argument	Type	Details
this: `ui.data.activedictionary`	ui.data.ActiveDictionary	The ui.data.ActiveDictionary instance.
`keyOrDict`	Object|String	Either the key of the property to set or a dictionary of key/value pairs to set on the object.
`value`	*Object, optional*	*The property's new value. This is required when the first argument is a key string.*

ui.data.ActiveList

An array-like container for data for use in UI components.

When a ui.data.ActiveList (e.g. Map.layers()) is updated, the component it belongs to is updated as well. For example, Map.layers().add(myLayer) will add myLayer as a layer on the map.

Usage	Returns
`ui.data.ActiveList(`*`list`*`)`	ui.data.ActiveList

Argument	Type	Details
`list`	*List, optional*	*An optional list to initialize with.*

ui.data.ActiveList.add

Appends an element to the list.

Returns this ui.data.ActiveList.

Usage	Returns
`ActiveList.add(el)`	ui.data.ActiveList

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`el`	Object	The element to add.

ui.data.ActiveList.forEach

Iterates over each element, calling the provided callback. The callback is called for each element like: callback(element, index).

Usage	Returns
`ActiveList.forEach(callback)`	

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`callback`	Function	

ui.data.ActiveList.get

Returns the element at the specified index.

Usage	Returns
`ActiveList.get(index)`	Object

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`index`	Number	The index of the element to return.

ui.data.ActiveList.getJsArray

Returns the list as a JS array.

Usage	Returns
`ActiveList.getJsArray()`	List

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.

ui.data.ActiveList.insert

Inserts an element at the specified index and shifts the rest of the list. If the specified index is greater than the length of the list, the element will be appended to the list.

Returns this ui.data.ActiveList.

Usage	Returns
`ActiveList.insert(index, el)`	ui.data.ActiveList

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`index`	Number	The index at which to insert the element.
`el`	Object	The element to insert.

ui.data.ActiveList.length

Returns the number of elements in the list.

Usage	Returns
`ActiveList.length()`	Number

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.

ui.data.ActiveList.remove

Removes the specified element from the list.

Returns the removed element or null if the element was not present in the list.

Usage	Returns
`ActiveList.remove(el)`	Object

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.

Argument	Type	Details
`el`	Object	The element to remove.

ui.data.ActiveList.reset

Replaces all elements in list with a new list or, if no list is provided, removes all elements from list.

Returns the elements in the list after the reset is applied.

Usage	Returns
`ActiveList.reset(`*`list`*`)`	List

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`list`	*List, optional*	*A list of elements.*

ui.data.ActiveList.set

Sets an element at the specified index. If the index exceeds that of the list's last element, the element will be added to the end of the list.

Returns this ui.data.ActiveList.

Usage	Returns
`ActiveList.set(index, el)`	ui.data.ActiveList

Argument	Type	Details
this: `ui.data.activelist`	ui.data.ActiveList	The ui.data.ActiveList instance.
`index`	Number	The index to overwrite.

Argument	Type	Details
`el`	Object	The element to set.

ui.root.add

Adds a widget to the root panel.

Returns the root panel.

Usage	Returns
`ui.root.add(widget)`	ui.Panel

Argument	Type	Details
`widget`	ui.Widget	The widget to be added.

ui.root.clear

Clears the root panel.

Usage	Returns
`ui.root.clear()`	

No arguments.

ui.root.getLayout

Returns the root panel's layout.

Usage	Returns
`ui.root.getLayout()`	ui.Panel.Layout

No arguments.

ui.root.insert

Inserts a widget into to the root panel at the specified index. Returns the root panel.

Usage	Returns
`ui.root.insert(index, widget)`	ui.Panel

Argument	Type	Details
`index`	Number	The index at which to insert the widget.
`widget`	ui.Widget	The widget to insert.

ui.root.onResize

Registers a callback that's fired when the script starts and whenever the browser window size changes. It will be passed an object with boolean fields "is_mobile", "is_tablet", "is_desktop", "is_portrait" and "is_landscape", and numeric fields "width" and "height".

These fields indicate whether a user's device is mobile, tablet or desktop, the device orientation (portrait or landscape), and the width and height of the window in pixels. See the Width and Height (dp) section of device metrics at https://material.io/resources/devices/.

Usage	Returns
`ui.root.onResize(callback)`	

Argument	Type	Details
`callback`	Function	The callback to fire after the window has been resized. The callback is passed an object with the information of the device.

ui.root.remove

Removes the given widget from the root panel, if it exists.

Returns the removed widget or null if the widget was not present in the root panel.

Usage	Returns
`ui.root.remove(widget)`	Object

Argument	Type	Details
`widget`	ui.Widget	The widget to remove.

ui.root.setKeyHandler

Sets a keydown event handler to the root panel with a non-predefined key. The handler is fired only once when a user presses the bound key command. The same key will be bound to the latest handler set to it.

Usage	Returns
`ui.root.setKeyHandler(keyCode, handler, `*`description`*`)`	

Argument	Type	Details
`keyCode`	List	A key code or an array of key codes. For example, ui.Key.A or [ui.Key.SHIFT, ui.Key.A].
`handler`	Function	The handler for the key command.
`description`	*String, optional*	*A short description that explains this key command. The description will be visible in the Shortcuts Menu.*

ui.root.setLayout

Sets the ui.root panel's layout.

Returns the root panel.

Usage	Returns
`ui.root.setLayout(layout)`	ui.Panel

Argument	Type	Details
`layout`	String|ui.Panel.Layout	The root panel's new layout.

ui.root.widgets

Returns the list of widgets currently in the root panel.

Usage	Returns
`ui.root.widgets()`	ui.data.ActiveList

No arguments.

ui.url.get

Returns the value of the given key from the URL fragment.

Usage	Returns
`ui.url.get(key, default)`	Boolean|Number|Object|String

Argument	Type	Details
`key`	String	The name of the parameter to read.
`default`	*Boolean|Number|String, optional*	*optional default value to return if no value is present.*

ui.url.set

Sets the value of the page's URL fragment. The fragment encodes a dictionary of keys and values. If a dictionary is supplied as the first argument, the key/value pairs in that dictionary will be encoded and replace the current URL fragment. If a key string is provided, then only that key (and its value, the second argument) are updated, and the rest of the URL fragment is unchanged.

Usage	Returns
`ui.url.set(keyOrDict, value)`	

Argument	Type	Details
`keyOrDict`	Dictionary	Either a key to update a single value in the URL fragment, or a dictionary of key/value pairs which will replace the existing URL fragment. Dictionary values must be of type string, number, or boolean.

Argument	Type	Details
`value`	*Boolean|Number|String, optional*	*The new value to associate with a single key. This is required when the first argument is a string and is otherwise ignored.*

ui.util.clear

Clears all state related to utility functions, including cancelling any active timeouts, intervals, debounces, etc.

Usage	Returns
`ui.util.clear()`	

No arguments.

ui.util.clearTimeout

Clears a timeout set via ui.util.setTimeout or ui.util.setInterval.

Usage	Returns
`ui.util.clearTimeout(timeoutKey)`	

Argument	Type	Details
`timeoutKey`	Number	The key to the timeout or interval to clear.

ui.util.debounce

Wraps a function to allow it to be called, at most, once for each sequence of calls fired repeatedly so long as they are fired less than a specified interval apart (in milliseconds). This can be used to reduce the number of invocations of an expensive function while ensuring it eventually runs.

Example use: For the callback to a change event on a ui.Checkbox. If the user clicks the checkbox repeatedly, only the last click of the checkbox will run the callback.

Returns the debounced function.

Usage	Returns
`ui.util.debounce(func, delay, scope)`	Function

Argument	Type	Details
`func`	Function	The function to debounce.
`delay`	Number	After the function is called once, the number of milliseconds to delay for an additional invocation of the function before allowing it to run.
`scope`	*Object, optional*	*Object in whose scope to call the function.*

ui.util.getCurrentPosition

Gets the user's current geographic position from the browser's geolocation service.

Usage	Returns
`ui.util.getCurrentPosition(success, error)`	

Argument	Type	Details
`success`	Function	A callback function that takes a ee.Geometry.Point object as its input parameter.
`error`	*Function, optional*	*An optional callback function that takes an error message as its input parameter.*

ui.util.rateLimit

Wraps a function to allow it to be called, at most, once per interval. If the wrapper function is called more than once, only the first call will go through, and no subsequent invocations will have an effect until the interval has elapsed. This can be used to ensure a function that is expensive to run executes immediately but doesn't execute repeatedly.

Example use: For the callback to a click on a ui.Button, in order to prevent the button from being accidentally double-clicked and the callback running twice.

Returns the rate-limited function.

Usage	Returns
`ui.util.rateLimit(func, delay, scope)`	Function

Argument	Type	Details
`func`	Function	Function to call.
`delay`	Number	After the function is called and executed, the number of milliseconds to delay before allowing an additional invocation of the function.
`scope`	*Object, optional*	*Object in whose scope to call the function.*

ui.util.setInterval

Repeatedly calls a function with a fixed time delay between each call.

Returns a key that can be passed to ui.util.clearTimeout to remove the timeout.

Usage	Returns
`ui.util.setInterval(func, delay)`	Number

Argument	Type	Details
`func`	Function	The function to run after the specified delay.
`delay`	Number	The time, in milliseconds (thousandths of a second), the timer should delay in between executions of the specified function.

ui.util.setTimeout

Calls a function after a fixed time delay.

Returns a key that can be passed to ui.util.clearTimeout to remove the timeout.

Usage	Returns
`ui.util.setTimeout(func, delay)`	Number

Argument	Type	Details
`func`	Function	The function to run at the specified interval.
`delay`	Number	The time, in milliseconds (thousandths of a second), the timer should delay before execution of the specified function.

ui.util.throttle

Wraps a function to allow it to be called, at most, twice per interval. If the wrapper function is called multiple times before the delay elapses, only the first and the last calls will go through.

Example use: For the callback to a slide event on a ui.Slider. The callback will run immediately, making the slide action feel responsive. The callback is also guaranteed to run after the user has finished interacting with the slider, ensuring that the final callback invocation has access to the slider's final value.

Returns the wrapped function.

Usage	Returns
`ui.util.throttle(func, delay, scope)`	Function

Argument	Type	Details
`func`	Function	The function to call.
`delay`	Number	The delay, in milliseconds, for the throttle. The function can only be called once after the initial invocation until after the delay has elapsed.
`scope`	*Object, optional*	*The object in whose scope to call the function.*