What's New in MapKit and MapKit JS

Alexander Jakobsen, MapKit Engineer
Nalini Shah, MapKit Engineer
WWDC Companion

Find Accommodation
Check out our partners for a great deal!

After Hours
Find the best places in San Jose

Concert in the Park
Thursday night at 8 PM. Be there!
Find Accommodation
Check out our partners for a great deal!

After Hours
Find the best places in San Jose

Concert in the Park
Thursday night at 8 PM. Be there!
WWDC Companion

Find Accommodation
Check out our partners for a great deal!

After Hours
Find the best places in San Jose

Concert in the Park
Thursday night at 8 PM. Be there!
Morton's The Steakhouse
177 Park Ave, San Jose, CA 95113, United States
WWDC Companion

Find Accommodation
Check out our partners for a great deal!

After Hours
Find the best places in San Jose

Concert in the Park
Thursday night at 8 PM. Be there!
Agenda
Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
Agenda

- Creating snapshots on the web
- Using the map in dark mode
- Filtering points of interest in map views
- Filtering search and autocompletion results
- Improving overlay performance
- Decoding and presenting GeoJSON
- Taking control of the map view camera
Steve Jobs Theater
Apple Building

Address
Apple Inc.
10501 N Tantau Ave
Cupertino, CA 95014
United States

Report an Issue
Maps Web Snapshots
Maps Web Snapshots

snapshot.apple-mapkit.com/api/v1/snapshot?
center=37.78,-122.42&size=640x300
Maps Web Snapshots

snapshot.apple-mapkit.com/api/v1/snapshot?
center=37.78,-122.42&size=640x300
Maps Web Snapshots

snapshot.apple-mapkit.com/api/v1/snapshot?
center=37.78,-122.42&size=640x300
snapshot.apple-mapkit.com/api/v1/snapshot?
center=37.78,-122.42&size=640x300&colorScheme=dark
Maps Web Snapshots

snapshot.apple-mapkit.com/api/v1/snapshot?
center=37.78,-122.42&size=640x300&colorScheme=dark
&signature=<yourSignatureHere>
WWDC 2019

See You Soon.

Attending WWDC

Make sure your week goes smoothly with these essential details. Find out how to get to the conference, review important reminders, and more.

Learn More
Create a Snapshot

Search for points of interest or directions to display on an Apple Maps Snapshot. Use the controls below to change characteristics of the snapshot such as dimension, language, scale, and color scheme, and click annotations or overlays to change their appearance.

Search
San Francisco, CA, United States

Width Height Language
500 500 English (United States)

- Dark Color Scheme
- Show Points of Interests
- Scale 2

Show Snapshot Reset

Enter your credentials to create Snapshot URLs for your own use.
developer.apple.com/mapkitjs

MapKit JS brings Apple Maps to your website. This new JavaScript library lets you add interactive maps to webpages — complete with annotations, overlays, and interfaces to Apple Maps services such as search and directions — to enable rich interactions. And with a free daily limit of 250,000 map views and 25,000 service calls, it has never been easier to deliver maps experiences to more of your users.
Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
Agenda

Creating snapshots on the web
Using the map in dark mode
Filtering points of interest in map views
Filtering search and autocompletion results
Improving overlay performance
Decoding and presenting GeoJSON
Taking control of the map view camera
MKMapSnapshotter
Maps without user interaction
// Create options
let options = MKMapSnapshotter.Options()
options.region = regionToSnapshot
options.size = CGSize(width: 300, height: 300)

// Alternative 1: Configure for view
options.traitCollection = snapshotView.traitCollection

// Alternative 2: Configure specific userInterfaceStyle
options.traitCollection = UITraitCollection(userInterfaceStyle: .dark)

// Create snapshot
let snapshotter = MKMapSnapshotter(options: options)
snapshotter.start { (snapshot, error) in
  // Handle snapshot if no error
}
// Create options
let options = MKMapSnapshotter.Options()
options.region = regionToSnapshot
options.size = CGSize(width: 300, height: 300)

// Alternative 1: Configure for view
options.traitCollection = snapshotView.traitCollection

// Alternative 2: Configure specific userInterfaceStyle
options.traitCollection = UITraitCollection(userInterfaceStyle: .dark)

// Create snapshot
let snapshotter = MKMapSnapshotter(options: options)
snapshotter.start { (snapshot, error) in
    // Handle snapshot if no error
}
// Create options
let options = MKMapSnapshotter.Options()
    options.region = regionToSnapshot
    options.size = CGSize(width: 300, height: 300)

// Alternative 1: Configure for view
options.traitCollection = snapshotView.traitCollection

// Alternative 2: Configure specific userInterfaceStyle
options.traitCollection = UITraitCollection(userInterfaceStyle: .dark)

// Create snapshot
let snapshotter = MKMapSnapshotter(options: options)
snapshotter.start { (snapshot, error) in
    // Handle snapshot if no error
}
// Create options
let options = MKMapSnapshotter.Options()
options.region = regionToSnapshot
options.size = CGSize(width: 300, height: 300)

// Alternative 1: Configure for view
options.traitCollection = snapshotView.traitCollection

// Alternative 2: Configure specific userInterfaceStyle
options.traitCollection = UITraitCollection(userInterfaceStyle: .dark)

// Create snapshot
let snapshotter = MKMapSnapshotter(options: options)
snapshotter.start { (snapshot, error) in
  // Handle snapshot if no error
}

// Create options
let options = MKMapSnapshotter.Options()
options.region = regionToSnapshot
options.size = CGSize(width: 300, height: 300)

// Alternative 1: Configure for view
options.traitCollection = snapshotView.traitCollection

// Alternative 2: Configure specific userInterfaceStyle
options.traitCollection = UITraitCollection(userInterfaceStyle: .dark)

// Create snapshot
let snapshotter = MKMapSnapshotter(options: options)
snapshotter.start { (snapshot, error) in
    // Handle snapshot if no error
}

// Create options
let options = MKMapSnapshotter.Options()
options.region = regionToSnapshot
options.size = CGSize(width: 300, height: 300)

// Alternative 1: Configure for view
options.traitCollection = snapshotView.traitCollection

// Alternative 2: Configure specific userInterfaceStyle
options.traitCollection = UITraitCollection(userInterfaceStyle: .dark)

// Create snapshot
let snapshotter = MKMapSnapshotter(options: options)
snapshotter.start { (snapshot, error) in
    // Handle snapshot if no error
}

// Create options

let options = MKMapSnapshotter.Options()
options.region = regionToSnapshot
options.size = CGSize(width: 300, height: 300)

// Alternative 1: Configure for view

optionstraitCollection = snapshotView.traitCollection

// Alternative 2: Configure specific userInterfaceStyle

optionstraitCollection = UITraitCollection(userInterfaceStyle: .dark)

// Create snapshot

let snapshotter = MKMapSnapshotter(options: options)
snapshotter.start { (snapshot, error) in
  // Handle snapshot if no error
}

// Create options
let options = MKMapSnapshotter.Options()
options.region = regionToSnapshot
options.size = CGSize(width: 300, height: 300)

// Alternative 1: Configure for view
options.traitCollection = snapshotView.traitCollection

// Alternative 2: Configure specific userInterfaceStyle
options.traitCollection = UITraitCollection(userInterfaceStyle: .dark)

// Create snapshot
let snapshotter = MKMapSnapshotter(options: options)

snapshotter.start { (snapshot, error) in
    // Handle snapshot if no error
}

Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
Agenda

Creating snapshots on the web
Using the map in dark mode
Filtering points of interest in map views
Filtering search and autocompletion results
Improving overlay performance
Decoding and presenting GeoJSON
Taking control of the map view camera
<table>
<thead>
<tr>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>airport</td>
</tr>
<tr>
<td>amusementPark</td>
</tr>
<tr>
<td>aquarium</td>
</tr>
<tr>
<td>atm</td>
</tr>
<tr>
<td>bakery</td>
</tr>
<tr>
<td>bank</td>
</tr>
<tr>
<td>beach</td>
</tr>
<tr>
<td>brewery</td>
</tr>
<tr>
<td>cafe</td>
</tr>
<tr>
<td>campground</td>
</tr>
<tr>
<td>carRental</td>
</tr>
<tr>
<td>.evCharger</td>
</tr>
<tr>
<td>.fireStation</td>
</tr>
<tr>
<td>.fitnessCenter</td>
</tr>
<tr>
<td>.foodMarket</td>
</tr>
<tr>
<td>.gasStation</td>
</tr>
<tr>
<td>.hospital</td>
</tr>
<tr>
<td>.hotel</td>
</tr>
<tr>
<td>.laundry</td>
</tr>
<tr>
<td>.library</td>
</tr>
<tr>
<td>.museum</td>
</tr>
<tr>
<td>.nationalPark</td>
</tr>
<tr>
<td>.nightlife</td>
</tr>
<tr>
<td>.park</td>
</tr>
<tr>
<td>.parking</td>
</tr>
<tr>
<td>.pharmacy</td>
</tr>
<tr>
<td>.playground</td>
</tr>
<tr>
<td>.police</td>
</tr>
<tr>
<td>.postOffice</td>
</tr>
<tr>
<td>.publicTransport</td>
</tr>
<tr>
<td>.restaurant</td>
</tr>
<tr>
<td>.restroom</td>
</tr>
<tr>
<td>.school</td>
</tr>
<tr>
<td>.stadium</td>
</tr>
<tr>
<td>.store</td>
</tr>
<tr>
<td>.theater</td>
</tr>
<tr>
<td>.university</td>
</tr>
<tr>
<td>.winery</td>
</tr>
<tr>
<td>.zoo</td>
</tr>
<tr>
<td>.movieTheater</td>
</tr>
<tr>
<td>.religiousSit</td>
</tr>
</tbody>
</table>
MKPointOfInterestFilter
MKPointOfInterestFilter

Include specific categories

Exclude specific categories

Exclude all categories
// Exclusion Filtering

let categories = [.hotel]
let filter = MKPointOfInterestFilter(excluding: categories)
mapView.pointOfInterestFilter = filter
// Inclusion Filtering

let categories = [ .restaurant, .cafe, .nightlife, .parking ]
let filter = MKPointOfInterestFilter(including: categories)
mapView.pointOfInterestFilter = filter
// Inclusion Filtering

let categories = [.restaurant, .cafe, .nightlife, .parking]
let filter = MKPointOfInterestFilter(including: categories)
mapView.pointOfInterestFilter = filter
// Inclusion Filtering

let categories = [.restaurant, .cafe, .nightlife, .parking]
let filter = MKPointOfInterestFilter(including: categories)
mapView.pointOfInterestFilter = filter
// Excluding All Points of Interest

let filter = MKPointOfInterestFilter.excludingAll
mapView.pointOfInterestFilter = filter
Point of Interest Filtering in MapKit JS

Coming this fall!

// Exclusion Filtering
let filter = mapkit.PointOfInterestFilter.excluding([ mapkit.PointOfInterestCategory.Hotel ]); 

// Inclusion Filtering
let filter = mapkit.PointOfInterestFilter.including([ mapkit.PointOfInterestCategory.Hotel ]); 

map.pointOfInterestFilter = filter;
Agenda

Creating snapshots on the web
Using the map in dark mode
Filtering points of interest in map views
Filtering search and autocompletion results
Improving overlay performance
Decoding and presenting GeoJSON
Taking control of the map view camera
Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
Morgan Hill, CA, United States
Morton's The Steakhouse
177 Park Ave, San Jose, CA 95113, United States
Morris Dailey Auditorium - San Jose State Uni...
1 Washington Sq, San Jose, CA 95192, United States
Morphe
2865 Stevens Creek Blvd, Santa Clara, CA 95050-8709, United States
Morrill Middle School
1970 Morrill Ave, San Jose, CA 95132, United States
Morrison Park Dr
San Jose, CA, United States
Morse St
San Jose, CA, United States
Morgan Hill, CA, United States

Morton's The Steakhouse
177 Park Ave, San Jose, CA 95113, United States

Morris Dailey Auditorium - San Jose State Uni...
1 Washington Sq, San Jose, CA 95192, United States

Morphe
2865 Stevens Creek Blvd, Santa Clara, CA 95050-8709, United States

Morrill Middle School
1970 Morrill Ave, San Jose, CA 95132, United States

Morrison Park Dr
San Jose, CA, United States

Morse St
San Jose, CA, United States
// Create point of interest filter
let categories = [.restaurant, .nightlife]
let pointOfInterestFilter = MKPointOfInterestFilter(including: categories)

// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.pointOfInterestFilter = pointOfInterestFilter
...
completer.queryFragment = query

// Search Request Setup
let request = MKLocalSearch.Request()
request.naturalLanguageQuery = query
request.pointOfInterestFilter = pointOfInterestFilter
...
request.start(completionHandler: completionHandler)
// Create point of interest filter
let categories = [.restaurant, .nightlife]
let pointOfInterestFilter = MKPointOfInterestFilter(including: categories)

// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.pointOfInterestFilter = pointOfInterestFilter
...
completer.queryFragment = query

// Search Request Setup
let request = MKLocalSearch.Request()
request.naturalLanguageQuery = query
request.pointOfInterestFilter = pointOfInterestFilter
...
request.start(completionHandler: completionHandler)
// Create point of interest filter
let categories = [.restaurant, .nightlife]
let pointOfInterestFilter = MKPointOfInterestFilter(including: categories)

// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.pointOfInterestFilter = pointOfInterestFilter
...
completer.queryFragment = query

// Search Request Setup
let request = MKLocalSearch.Request()
request.naturalLanguageQuery = query
request.pointOfInterestFilter = pointOfInterestFilter
...
request.start(completionHandler: completionHandler)
// Create point of interest filter
let categories = [.restaurant, .nightlife]
let pointOfInterestFilter = MKPointOfInterestFilter(including: categories)

// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.pointOfInterestFilter = pointOfInterestFilter
...
completer.queryFragment = query

// Search Request Setup
let request = MKLocalSearch.Request()
request.naturalLanguageQuery = query
request.pointOfInterestFilter = pointOfInterestFilter
...
request.start(completionHandler: completionHandler)
Morgan Hill, CA, United States
Morton's The Steakhouse
177 Park Ave, San Jose, CA 95113, United States

Morrison Park Dr
San Jose, CA, United States

Morse St
San Jose, CA, United States

Morse St
Santa Clara, CA, United States
// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.delegate = self
completer.pointOfInterestFilter = pointOfInterestFilter
completer.filterType = .locationsOnly

...

// User input has changed
completer.queryFragment = query
// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.delegate = self
completer.pointOfInterestFilter = pointOfInterestFilter

completer.filterType = .locationsOnly

...

// User input has changed
completer.queryFragment = query
ResultType

MKLocalSearchCompleter
**ResultType**
MKLocalSearchCompleter

.address
Example: 10600 N Tantau Ave, Cupertino, CA 95014

.pointOfInterest
Example: Apple Park Visitor Center

.queries
Example: coffee
ResultType
MKLocalSearch.Request

.address
Example: 10600 N Tantau Ave, Cupertino, CA 95014

.pointOfInterest
Example: Apple Park Visitor Center
Result Type Filtering

// Search Request Setup
let request = MKLocalSearch.Request()
request.naturalLanguageQuery = query
request.pointOfInterestFilter = pointOfInterestFilter
request.resultTypes = .pointOfInterest

// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.delegate = self
completer.pointOfInterestFilter = pointOfInterestFilter
completer.resultTypes = .pointOfInterest
Result Type Filtering

// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.delegate = self
completer.pointOfInterestFilter = pointOfInterestFilter

completer.resultTypes = .pointOfInterest

// Search Request Setup
let request = MKLocalSearch.Request()
request.naturalLanguageQuery = query
request.pointOfInterestFilter = pointOfInterestFilter

request.resultTypes = .pointOfInterest
Result Type Filtering

// Autocompletion Setup
let completer = MKLocalSearchCompleter()
completer.delegate = self
completer.pointOfInterestFilter = pointOfInterestFilter
completer.resultTypes = .pointOfInterest

// Search Request Setup
let request = MKLocalSearch.Request()
request.naturalLanguageQuery = query
request.pointOfInterestFilter = pointOfInterestFilter
request.resultTypes = .pointOfInterest
MKMapItem

Location
Address
Name
Phone number
URL
MKMapItem

Location
Address
Name
Phone number
URL
Point of interest category
MKMapItem
Point of interest category

Search Request “ABC” ➔ MKLocalSearch
MKMapItem
Point of interest category

Search Request
“ABC”

MKLocalSearch

Al’s Beet Canteen
.restaurant

ABC Brewing
.brewery
MKMapItem
Point of interest category

Search Request
"ABC"
\*brewery*

ABC Brewing
\*brewery*
MKMapItem
Point of interest category

Search Request
"ABC"
.restaurant

MKLocalSearch

Al's Beet Canteen .restaurant

ABC Brewing .brewery
MKMapItem
Point of interest category

MKLocalSearch

Search Request
"ABC"
.restaurant

Al's Beet Canteen
.restaurant

ABC Brewing
.brewery
Search and Autocomplete Filtering in MapKit JS

Coming this fall!

// Filtering points of interest
let filter = mapkit.PointOfInterestFilter.including([ mapkit.PointOfInterestCategory.Hotel ]); search.pointOfInterestFilter = filter;

// Filtering result types
search.includeAddresses = false;
search.includePointsOfInterest = true;
search.includeQueries = false;
Demo

Improving search and completion results

Nalini Shah, MapKit Engineer
Agenda

Creating snapshots on the web
Using the map in dark mode
Filtering points of interest in map views
Filtering search and autocompletion results
Improving overlay performance
Decoding and presenting GeoJSON
Taking control of the map view camera
Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
Improving Polygon and Polyline Performance
Improving Polygon and Polyline Performance

MKMultiPolygon
- Groups polygons of the same style

MKMultiPolyline
- Groups polylines of the same style
Improving Polygon and Polyline Performance
Improving Polygon and Polyline Performance

MKMultiPolygonRenderer
• Renders a group of polygons of the same style

MKMultiPolylineRenderer
• Renders a group of polylines of the same style
// Adding polygons to a map view

let stageCoordinates = [
    CLLocationCoordinate2D(latitude: 37.332536, longitude: -121.889562),
    CLLocationCoordinate2D(latitude: 37.332312, longitude: -121.890037),
    CLLocationCoordinate2D(latitude: 37.332082, longitude: -121.889863),
    CLLocationCoordinate2D(latitude: 37.332310, longitude: -121.889393)
]

let stage = MKPolygon(coordinates: stageCoordinates, count: 4)

// Creation of other polygons omitted...

mapView.addOverlays([stage, mainBar, bar, tacos, burgers, falafel, restroom])
// Adding polygons to a map view

let stageCoordinates = [
    CLLocationCoordinate2D(latitude: 37.332536, longitude: -121.889562),
    CLLocationCoordinate2D(latitude: 37.332312, longitude: -121.890037),
    CLLocationCoordinate2D(latitude: 37.332082, longitude: -121.889863),
    CLLocationCoordinate2D(latitude: 37.332310, longitude: -121.889393)
]

let stage = MKPolygon(coordinates: stageCoordinates, count: 4)

// Creation of other polygons omitted...

mapView.addOverlays([stage, mainBar, bar, tacos, burgers, falafel, restroom])
// Adding polygons to a map view

let stageCoordinates = [
    CLLocationCoordinate2D(latitude: 37.332536, longitude: -121.889562),
    CLLocationCoordinate2D(latitude: 37.332312, longitude: -121.890037),
    CLLocationCoordinate2D(latitude: 37.332082, longitude: -121.889863),
    CLLocationCoordinate2D(latitude: 37.332310, longitude: -121.889393)
]

let stage = MKPolygon(coordinates: stageCoordinates, count: 4)

// Creation of other polygons omitted...

mapView.addOverlays([[stage, mainBar, bar, tacos, burgers, falafel, restroom]])
// Adding a multipolygon to a map view

let stageCoordinates = [
    CLLocationCoordinate2D(latitude: 37.332536, longitude: -121.889562),
    CLLocationCoordinate2D(latitude: 37.332312, longitude: -121.890037),
    CLLocationCoordinate2D(latitude: 37.332082, longitude: -121.889863),
    CLLocationCoordinate2D(latitude: 37.332310, longitude: -121.889393)
]

let stage = MKPolygon(coordinates: stageCoordinates, count: 4)

// Creation of other polygons omitted...

let venue = MKMultiPolygon(polygons: [stage, mainBar, bar, tacos, burgers, falafel, restroom])
mapView.addOverlay(venue)
// Adding a multipolygon to a map view

let stageCoordinates = [
    CLLocationCoordinate2D(latitude: 37.332536, longitude: -121.889562),
    CLLocationCoordinate2D(latitude: 37.332312, longitude: -121.890037),
    CLLocationCoordinate2D(latitude: 37.332082, longitude: -121.889863),
    CLLocationCoordinate2D(latitude: 37.332310, longitude: -121.889393)
]

let stage = MKPolygon(coordinates: stageCoordinates, count: 4)

// Creation of other polygons omitted...

let venue = MKMultiPolygon(polygons: [stage, mainBar, bar, tacos, burgers, falafel, restroom])
mapView.addOverlay(venue)
// Adding a multipolygon to a map view

let stageCoordinates = [
    CLLocationCoordinate2D(latitude: 37.332536, longitude: -121.889562),
    CLLocationCoordinate2D(latitude: 37.332312, longitude: -121.890037),
    CLLocationCoordinate2D(latitude: 37.332082, longitude: -121.889863),
    CLLocationCoordinate2D(latitude: 37.332310, longitude: -121.889393)
]

let stage = MKPolygon(coordinates: stageCoordinates, count: 4)

// Creation of other polygons omitted...

let venue = MKMultiPolygon(polygons: [stage, mainBar, bar, tacos, burgers, falafel, restroom])

mapView.addOverlay(venue)
// Providing a multipolygon renderer

func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
    if let multiPolygon = overlay as? MKMultiPolygon {
        let renderer = MKMultiPolygonRenderer(polygon: multiPolygon)
        renderer.fillColor = UIColor(named: "VenueFill")
        renderer.strokeColor = UIColor(named: "VenueStroke")
        renderer.lineWidth = 2.0

        return renderer
    }

    return MKOverlayRenderer(overlay: overlay)
}
// Providing a multipolygon renderer

func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {

    if let multiPolygon = overlay as? MKMultiPolygon {
        let renderer = MKMultiPolygonRenderer(polygon: multiPolygon)
        renderer.fillColor = UIColor(named: "VenueFill")
        renderer.strokeColor = UIColor(named: "VenueStroke")
        renderer.lineWidth = 2.0

        return renderer
    }

    return MKOverlayRenderer(overlay: overlay)
}
func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {

    if let multiPolygon = overlay as? MKMultiPolygon {
        let renderer = MKMultiPolygonRenderer(polygon: multiPolygon)
        renderer.fillColor = UIColor(named: "VenueFill")
        renderer.strokeColor = UIColor(named: "VenueStroke")
        renderer.lineWidth = 2.0

        return renderer
    }

    return MKOverlayRenderer(overlay: overlay)
}
// Opt out of vector rendering

open class MKOverlayPathRenderer: MKOverlayRenderer {

    open var shouldRasterize: Bool

}
/ * Opt out of vector rendering *

```swift
open class MKOverlayPathRenderer: MKOverlayRenderer {
    open var shouldRasterize: Bool
}
```
Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
Agenda

Creating snapshots on the web
Using the map in dark mode
Filtering points of interest in map views
Filtering search and autocompletion results
Improving overlay performance
Decoding and presenting GeoJSON
Taking control of the map view camera
/ Example GeoJSON (RFC 7946)

{
    "type": "Feature",
    "identifier": "SampleFeatureID",
    "geometry": {
        "type": "Point",
        "coordinates": [
            [ -121.890354, 37.332744 ]
        ]
    },
    "properties": {
        "name": "Stage"
    }
}
// Example GeoJSON (RFC 7946)

{
  "type": "Feature",
  "identifier": "SampleFeatureID",
  "geometry": {
    "type": "Point",
    "coordinates": [
      [-121.890354, 37.332744]
    ]
  },
  "properties": {
    "name": "Stage"
  }
}
// Example GeoJSON (RFC 7946)

{
    "type": "Feature",
    "identifier": "SampleFeatureID",
    "geometry": {
        "type": "Point",
        "coordinates": [
            [ -121.890354, 37.332744 ]
        ]
    },
    "properties": {
        "name": "Stage"
    }
}
// Example GeoJSON (RFC 7946)
{
   "type": "Feature",
   "identifier": "SampleFeatureID",
   "geometry": {
      "type": "Point",
      "coordinates": [
         [ -121.890354, 37.332744 ]
      ]
   },
   "properties": {
      "name": "Stage"
   }
}
// Example GeoJSON (RFC 7946)

{
    "type": "Feature",
    "identifier": "SampleFeatureID",
    "geometry": {
        "type": "Point",
        "coordinates": [
            [ -121.890354, 37.332744 ]
        ]
    },
    "properties": {
        "name": "Stage"
    }
}
MKGeoJSONFeature

NEW
MKGeoJSONFeature

Identifier

Geometry

Properties data
MKGeoJSONDecoder
Decoding GeoJSON data into MapKit objects
MKGeoJSONDecoder
Decoding GeoJSON data into MapKit objects
MKGeoJSONDecoder
Decoding GeoJSON data into MapKit objects

JSON → MKGeoJSONDecoder
MKGeoJSONDecoder
Decoding GeoJSON data into MapKit objects
MKGeoJSONDecoder
Decoding GeoJSON data into MapKit objects

NEW

JSON

MKGeoJSONDecoder

MapKit Geometry

MapKit Geometry

MapKit Geometry
// Example GeoJSON (RFC 7946)

{
    "type": "Feature",
    "identifier": "SampleFeatureID",
    "geometry": {
        "type": "Point",
        "coordinates": [
            [ -121.890354, 37.332744 ]
        ]
    },
    "properties": {
        "name": "Stage"
    }
}
// Example GeoJSON (RFC 7946)

{
    "type": "Feature",
    "identifier": "SampleFeatureID",
    "geometry": {
        MKPointAnnotation
    },
    "properties": {
        MKGeoJSONDecoder
        "name": "Stage"
    }
}
Example GeoJSON (RFC 7946):

```
{
  "type": "Feature",
  "identifier": "SampleFeatureID",
  "geometry": {
    "type": "Point",
    "coordinates": [-121.890354, 37.332744]
  },
  "properties": {
    "name": "Stage"
  }
}
```
Mapping GeoJSON to MapKit
## Mapping GeoJSON to MapKit

<table>
<thead>
<tr>
<th>GeoJSON</th>
<th>MapKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>MKPointAnnotation</td>
</tr>
<tr>
<td>MultiPoint</td>
<td>MKMultiPoint</td>
</tr>
<tr>
<td>LineString</td>
<td>MKPolyline</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>MKMultiPolyline</td>
</tr>
<tr>
<td>Polygon</td>
<td>MKPolygon</td>
</tr>
<tr>
<td>MultiPolygon</td>
<td>MKMultiPolygon</td>
</tr>
</tbody>
</table>
// Decoding GeoJSON

do {
    let decoder = MKGeoJSONDecoder()
    let decodedObjects = try decoder.decode(venueData)

    for object in decodedObjects {
        if let feature = object as? MKGeoJSONFeature {
            parse(feature: feature)
        } else if let geometry = object as? (MKShape & MKGeoJSONObject) {
            parse(geometry: geometry)
        }
    }
} catch {
    print("Unexpected error: \(error).")
}
// Decoding GeoJSON

do {
    let decoder = MKGeoJSONDecoder()
    let decodedObjects = try decoder.decode(venueData)

    for object in decodedObjects {
        if let feature = object as? MKGeoJSONFeature {
            parse(feature: feature)
        } else if let geometry = object as? (MKShape & MKGeoJSONObject) {
            parse(geometry: geometry)
        }
    }
} catch {
    print("Unexpected error: \(error).")
}
// Decoding GeoJSON

do {
    let decoder = MKGeoJSONDecoder()
    let decodedObjects = try decoder.decode(venueData)

    for object in decodedObjects {
        if let feature = object as? MKGeoJSONFeature {
            parse(feature: feature)
        } else if let geometry = object as? (MKShape & MKGeoJSONObject) {
            parse(geometry: geometry)
        }
    }
} catch {
    print("Unexpected error: \(error).")
}
// Decoding GeoJSON

do {
    let decoder = MKGeoJSONDecoder()
    let decodedObjects = try decoder.decode(venueData)

    for object in decodedObjects {
        if let feature = object as? MKGeoJSONFeature {
            parse(feature: feature)
        } else if let geometry = object as? (MKShape & MKGeoJSONObject) {
            parse(geometry: geometry)
        }
    }
} catch {
    print("Unexpected error: \(error).")
}
// Decoding GeoJSON

do {
    let decoder = MKGeoJSONDecoder()
    let decodedObjects = try decoder.decode(venueData)

    for object in decodedObjects {
        if let feature = object as? MKGeoJSONFeature {
            parse(feature: feature)
        } else if let geometry = object as? (MKShape & MKGeoJSONObject) {
            parse(geometry: geometry)
        }
    }
} catch {
    print("Unexpected error: \(error).")
}
// Parsing a feature's 'properties' data

private func parse(feature: MKGeoJSONFeature) {

    if let point = feature.geometry.first as? MKPointAnnotation {

        if let properties = feature.properties {

            let decoder = JSONDecoder()
            if let dictionary = try? decoder.decode([String: String].self, from: properties) {
                point.title = dictionary["name"]
            }
        }

        annotations.append(point)
    }
}
private func parse(feature: MKGeoJSONFeature) {

if let point = feature.geometry.first as? MKPointAnnotation {

if let properties = feature.properties {

let decoder = JSONDecoder()
if let dictionary = try? decoder.decode([String: String].self, from: properties) {
    point.title = dictionary["name"]
}

annotations.append(point)
}
}
// Parsing a feature's 'properties' data

private func parse(feature: MKGeoJSONFeature) {

    if let point = feature.geometry.first as? MKPointAnnotation {
        if let properties = feature.properties {

            let decoder = JSONDecoder()
            if let dictionary = try? decoder.decode([String: String].self, from: properties) {
                point.title = dictionary["name"]
            }

        }

        annotations.append(point)
    }
}
private func parse(feature: MKGeoJSONFeature) {

    if let point = feature.geometry.first as? MKPointAnnotation {

        if let properties = feature.properties {

            let decoder = JSONDecoder()
            if let dictionary = try? decoder.decode([String: String].self, from: properties) {
                point.title = dictionary["name"]
            }
        }

        annotations.append(point)
    }
}
/ Parsing a feature's 'properties' data

private func parse(feature: MKGeoJSONFeature) {

    if let point = feature.geometry.first as? MKPointAnnotation {

        if let properties = feature.properties {

            let decoder = JSONDecoder()
            if let dictionary = try? decoder.decode([String: String].self, from: properties) {
                point.title = dictionary["name"]
            }
        }
    }

    annotations.append(point)
}
}
// Parsing unknown 'properties' data

if let properties = feature.properties {

    if let json = try? JSONSerialization.jsonObject(with: properties, options: []) {

        if let dictionary = json as? [String: Any] {
            // Inspect Dictionary
        } else if let array = json as? [Any] {
            // Inspect array
        }
    }
}
// Parsing unknown 'properties' data

if let properties = feature.properties {

    if let json = try? JSONSerialization.jsonObject(with: properties, options: []) {

        if let dictionary = json as? [String: Any] {
            // Inspect Dictionary
        } else if let array = json as? [Any] {
            // Inspect array
        }
    }
}
// Parsing unknown 'properties' data

if let properties = feature.properties {

    if let json = try? JSONSerialization.jsonObject(with: properties, options: []) {

        if let dictionary = json as? [String: Any] {
            // Inspect Dictionary
        } else if let array = json as? [Any] {
            // Inspect array
        }
    }
}
Mapping GeoJSON to MapKit JS
# Mapping GeoJSON to MapKit JS

<table>
<thead>
<tr>
<th>GeoJSON</th>
<th>MapKit JS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>MarkerAnnotation</td>
</tr>
<tr>
<td>MultiPoint</td>
<td>ItemCollection</td>
</tr>
<tr>
<td>LineString</td>
<td>PolylineOverlay</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>ItemCollection</td>
</tr>
<tr>
<td>Polygon</td>
<td>PolygonOverlay</td>
</tr>
<tr>
<td>MultiPolygon</td>
<td>ItemCollection</td>
</tr>
<tr>
<td>Feature</td>
<td>ItemCollection, MarkerAnnotation, Overlay</td>
</tr>
</tbody>
</table>
// Importing GeoJSON in MapKit JS

var mapKitItems = mapkit.importGeoJSON(
   "https://example.com/features.geojson",
   geoJsonDelegate
);

map.showItems(mapKitItems);
Demo
Decoding GeoJSON
Indoor Mapping Data Format
Indoor Mapping Data Format

Built on top of GeoJSON
Indoor Mapping Data Format

Built on top of GeoJSON

Complete set of features to represent a venue

Adding Indoor Maps to your App and Website

Introducing the Indoor Map Program
Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
MKMapView.CameraBoundary

Constrains the map view’s center point
MKMapView.CameraBoundary

Constrains the map view’s center point
Applying a Camera Boundary

// Alternative 1: Create using an MKCoordinateRegion
let region = MKCoordinateRegion(center: center, latitudinalMeters: 80, longitudinalMeters: 90)
mapView.cameraBoundary = MKMapView.CameraBoundary(coordinateRegion: region)

// Alternative 2: Create using an MKMapRect
let mapRect = MKMapRect(origin: origin, size: size)
mapView.cameraBoundary = MKMapView.CameraBoundary(mapRect: mapRect)
Viewport
Applying a Camera Boundary in MapKit JS

// Alternative 1: Apply a CoordinateRegion as camera boundary
map.cameraBoundary = mapkit.CoordinateRegion(coordinate, span);

// Alternative 2: Apply a MapRect as camera boundary
map.cameraBoundary = mapkit.MapRect(x, y, width, height);
Center Coordinate

Altitude
1000 m
Center Coordinate

Altitude

1000 m
MKMapView.CameraZoomRange

Defines a minimum and maximum center coordinate distance
Applying a Camera Zoom Range

// Alternative 1: Create using both min and max center coordinate distances
mapView.cameraZoomRange = MKMapView.CameraZoomRange(minCenterCoordinateDistance: 500,
maxCenterCoordinateDistance: 2000)

// Alternative 2: Create using only a min center coordinate distance
mapView.cameraZoomRange = MKMapView.CameraZoomRange(minCenterCoordinateDistance: 500)

// Alternative 3: Create using only a max center coordinate distance
mapView.cameraZoomRange = MKMapView.CameraZoomRange(maxCenter CoordinateDistance: 500)
Camera Zoom Range

Center Coordinate Distance

Min Max
Applying a Camera Zoom Range in MapKit JS

// Alternative 1: Create with both min and max camera distance
map.cameraZoomRange = new mapkit.CameraZoomRange(500, 2000);

// Alternative 2: Create with only min camera distance
map.cameraZoomRange = new mapkit.CameraZoomRange({minCameraDistance: 500});

// Alternative 3: Create with only max camera distance
map.cameraZoomRange = new mapkit.CameraZoomRange({maxCameraDistance: 2000});
Demo

Applying camera boundaries and zoom ranges

Nalini Shah, MapKit Engineer
Agenda

- Creating snapshots on the web
- Using the map in dark mode
- Filtering points of interest in map views
- Filtering search and autocompletion results
- Improving overlay performance
- Decoding and presenting GeoJSON
- Taking control of the map view camera
Agenda

• Creating snapshots on the web
• Using the map in dark mode
• Filtering points of interest in map views
• Filtering search and autocompletion results
• Improving overlay performance
• Decoding and presenting GeoJSON
• Taking control of the map view camera
Summary
Replace maps with snapshots if interaction is not needed
Summary

Replace maps with snapshots if interaction is not needed

Apply point of interest filters to tailor your maps to your needs
Summary

Replace maps with snapshots if interaction is not needed

Apply point of interest filters to tailor your maps to your needs

Filter search and autocompletion results for improved relevance
Summary

Replace maps with snapshots if interaction is not needed

Apply point of interest filters to tailor your maps to your needs

Filter search and autocompletion results for improved relevance

Group your polygons and polylines into multipolygons and multipolylines
Summary

Replace maps with snapshots if interaction is not needed
Apply point of interest filters to tailor your maps to your needs
Filter search and autocompletion results for improved relevance
Group your polygons and polylines into multipolygons and multipolylines
Take advantage of MapKit and MapKit JS’s GeoJSON support
Summary

Replace maps with snapshots if interaction is not needed

Apply point of interest filters to tailor your maps to your needs

Filter search and autocompletion results for improved relevance

Group your polygons and polylines into multipolygons and multipolylines

Take advantage of MapKit and MapKit JS’s GeoJSON support

Use camera boundaries and zoom ranges to focus on what really matters
More Information

developer.apple.com/wwdc19/236

<table>
<thead>
<tr>
<th>Title</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding Indoor Maps to your App and Website</td>
<td>Friday, 2:00</td>
</tr>
<tr>
<td>MapKit, MapKit JS, and Indoor Location Lab</td>
<td>Friday, 3:00</td>
</tr>
<tr>
<td>Introducing the Indoor Map Program</td>
<td>WWDC 2019</td>
</tr>
</tbody>
</table>