```
import ARKit.ARAnchor
import ARKit.ARAppClipCodeAnchor
import ARKit.ARBody2D
import ARKit.ARBodyAnchor
import ARKit.ARCamera
import ARKit.ARCoachingOverlayView
import ARKit.ARCollaborationData
import ARKit.ARConfiguration
import ARKit.ARDepthData
import ARKit.AREnvironmentProbeAnchor
import ARKit.ARError
import ARKit.ARFaceAnchor
import ARKit.ARFaceGeometry
import ARKit.ARFrame
import ARKit.ARGeoAnchor
import ARKit.ARGeoTrackingTypes
import ARKit.ARHitTestResult
import ARKit.ARImageAnchor
import ARKit.ARKitCore
import ARKit.ARKitFoundation
import ARKit.ARKitUI
import ARKit.ARLightEstimate
import ARKit.ARMatteGenerator
import ARKit.ARMeshAnchor
import ARKit.ARMeshGeometry
import ARKit.ARObjectAnchor
import ARKit.ARParticipantAnchor
import ARKit.ARPlaneAnchor
import ARKit.ARPlaneDetectionTypes
import ARKit.ARPlaneGeometry
import ARKit.ARPointCloud
import ARKit.ARQuickLookPreviewItem
import ARKit.ARRaycastQuery
```

```swift
import ARKit.ARRaycastResult
import ARKit.ARReferenceImage
import ARKit.ARReferenceObject
import ARKit.ARSCNFaceGeometry
import ARKit.ARSCNPlaneGeometry
import ARKit.ARSCNView
import ARKit.ARSKView
import ARKit.ARSession
import ARKit.ARSkeleton
import ARKit.ARSkeletonDefinition
import ARKit.ARTrackedRaycast
import ARKit.ARTrackingStatusTypes
import ARKit.ARVideoFormat
import ARKit.ARWorldMap
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims
import simd

@available(iOS 12.0, *)
extension ARPlaneAnchor {

    /**
     A value describing the
classification of a plane anchor.
     */
    public enum Classification : Sendable
{

        public enum Status : Sendable {

            /** Plane classification is
currently unavailable. */
```

```
            case notAvailable

            /** ARKit has not yet
determined the classification of this
plane. */
            case undetermined

            /** ARKit is confident the
plane is not any of the known classes. */
            case unknown

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
ARPlaneAnchor.Classification.Status, b:
ARPlaneAnchor.Classification.Status) ->
Bool

            /// Hashes the essential
components of this value by feeding them
into the
```

```
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
```

```
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        /** The classification is not any
of the known classes. */
        case
none(ARPlaneAnchor.Classification.Status)

        case wall

        case floor

        case ceiling

        case table

        case seat
        case window

        case door
    }

    /**
     Classification of the plane.
     */
    public var classification:
ARPlaneAnchor.Classification { get }
}

@available(iOS 11.0, *)
extension ARPointCloud {

    /**
     The 3D points comprising the point
cloud.
     */
    @nonobjc public var points:
[simd_float3] { get }

    /**
     The 3D point identifiers comprising
the point cloud.
     */
    @nonobjc public var identifiers:
[UInt64] { get }
}

@available(iOS 14.0, *)
extension ARConfidenceLevel : Comparable
{
    /// Returns a Boolean value
indicating whether the value of the first
    /// argument is less than that of the
second argument.
    ///
    /// This function is the only
requirement of the `Comparable` protocol.
The
    /// remainder of the relational
operator functions are implemented by the
    /// standard library for any type
that conforms to `Comparable`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func < (lhs:
ARConfidenceLevel, rhs:
ARConfidenceLevel) -> Bool
}

@available(iOS 13.4, *)
extension ARGeometrySource {

    /**
     Access an element (3D vector) of the
geometry source where the geometry
source's .format is
MTLVertexFormat.float3.
     @discussion Using this subscript
operator leads to a crash if the geometry
```

```swift
source has a .format other than
MTLVertexFormat.float3.
     @param index: Index of the element
to access
     @return A tuple of three floats
    */
    @available(iOS 14.0, *)
    @nonobjc public subscript(index:
Int32) -> (Float, Float, Float) { get }

    /**
     Access an element of the geometry
source where the geometry
source's .format is
MTLVertexFormat.uchar.
     @discussion Using this subscript
operator leads to a crash if the geometry
source has a .format other than
MTLVertexFormat.uchar.
     @param index: Index of the element
to access
     @return An unsigned char value
    */
    @available(iOS 14.0, *)
    @nonobjc public subscript(index:
Int32) -> CUnsignedChar { get }
}

@available(iOS 13.4, *)
extension ARGeometryElement {

    /**
     Access an array of values of the
```

```
geometry element where the geometry
element's .bytesPerIndex equals
MemoryLayout<Int32>.size.
     @discussion Using this subscript
operator leads to a crash if the geometry
element's .bytesPerIndex does not equal
MemoryLayout<Int32>.size.
     @param index: Index of the array of
values to access
     @return An array of Int32
    */
    @available(iOS 14.0, *)
    @nonobjc public subscript(index: Int)
-> [Int32] { get }
}

@available(iOS 11.3, *)
extension ARPlaneGeometry {

    /**
     The mesh vertices of the geometry.
     */
    @nonobjc public var vertices:
[simd_float3] { get }

    /**
     The texture coordinates of the
geometry.
     */
    @nonobjc public var
textureCoordinates: [vector_float2] { get
}
```

```
    /**
     The triangle indices of the
geometry.
     */
    @nonobjc public var triangleIndices:
[Int16] { get }

    /**
     The vertices of the geometry's
outermost boundary.
     */
    @nonobjc public var boundaryVertices:
[simd_float3] { get }
}

@available(iOS 12.0, *)
extension ARSCNView {

    @MainActor @nonobjc @preconcurrency
public func unprojectPoint(_ point:
CGPoint, ontoPlane planeTransform:
simd_float4x4) -> simd_float3?
}

@available(iOS 14.0, *)
extension ARGeoAnchor {

    @nonobjc public convenience
init(coordinate: CLLocationCoordinate2D,
altitude: CLLocationDistance? = nil)

    @nonobjc public convenience
init(name: String, coordinate:
```

```
CLLocationCoordinate2D, altitude:
CLLocationDistance? = nil)

    @nonobjc public var altitude:
CLLocationDistance? { get }
}

@available(iOS 13.0, *)
extension ARSkeletonDefinition {

    @nonobjc public var parentIndices:
[Int] { get }

    @nonobjc public func index(for
jointName: ARSkeleton.JointName) -> Int
}

@available(iOS 13.0, *)
extension ARSkeleton3D {

    @nonobjc public var
jointModelTransforms: [simd_float4x4] {
get }

    @nonobjc public var
jointLocalTransforms: [simd_float4x4] {
get }

    @nonobjc public func
modelTransform(for jointName:
ARSkeleton.JointName) -> simd_float4x4?

    @nonobjc public func
```

```
localTransform(for jointName:
ARSkeleton.JointName) -> simd_float4x4?
}

@available(iOS 13.0, *)
extension ARSkeleton2D {

    @nonobjc public var jointLandmarks:
[simd_float2] { get }

    @nonobjc public func landmark(for
jointName: ARSkeleton.JointName) ->
simd_float2?
}

@available(iOS 11.0, *)
extension ARFaceGeometry {

    /**
     The mesh vertices of the geometry.
     */
    @nonobjc public var vertices:
[simd_float3] { get }

    /**
     The texture coordinates of the
geometry.
     */
    @nonobjc public var
textureCoordinates: [vector_float2] { get
}

    /**
```

```
     The triangle indices of the
geometry.
     */
    @nonobjc public var triangleIndices:
[Int16] { get }
}

@available(iOS 11.0, *)
extension ARCamera {

    /**
     A value describing the camera's
tracking state.
     */
    @frozen public enum TrackingState :
Sendable {

        public enum Reason : Sendable {

            /** Tracking is limited due
to initialization in progress. */
            case initializing

            /** Tracking is limited due
to a excessive motion of the camera. */
            case excessiveMotion

            /** Tracking is limited due
to a lack of features visible to the
camera. */
            case insufficientFeatures

            /** Tracking is limited due
```

```
to a relocalization in progress. */
            @available(iOS 11.3, *)
            case relocalizing

            /// Returns a Boolean value
indicating whether two values are equal.
            ///
            /// Equality is the inverse
of inequality. For any values `a` and
`b`,
            /// `a == b` implies that
`a != b` is `false`.
            ///
            /// - Parameters:
            ///   - lhs: A value to
compare.
            ///   - rhs: Another value to
compare.
            public static func == (a:
ARCamera.TrackingState.Reason, b:
ARCamera.TrackingState.Reason) -> Bool

            /// Hashes the essential
components of this value by feeding them
into the
            /// given hasher.
            ///
            /// Implement this method to
conform to the `Hashable` protocol. The
            /// components used for
hashing must be the same as the
components compared
            /// in your type's `==`
```

```
operator implementation. Call
`hasher.combine(_:)`
            /// with each of these
components.
            ///
            /// - Important: In your
implementation of `hash(into:)`,
            ///   don't call `finalize()`
on the `hasher` instance provided,
            ///   or replace it with a
different instance.
            ///   Doing so may become a
compile-time error in the future.
            ///
            /// - Parameter hasher: The
hasher to use when combining the
components
            ///   of this instance.
            public func hash(into hasher:
inout Hasher)

            /// The hash value.
            ///
            /// Hash values are not
guaranteed to be equal across different
executions of
            /// your program. Do not save
hash values to use during a future
execution.
            ///
            /// - Important: `hashValue`
is deprecated as a `Hashable`
requirement. To
```

```
            ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
            ///   The compiler provides
an implementation for `hashValue` for
you.
            public var hashValue: Int {
get }
        }

        /** Tracking is not available. */
        case notAvailable

        /** Tracking is limited. See
tracking reason for details. */
        case
limited(ARCamera.TrackingState.Reason)

        /** Tracking is normal. */
        case normal
    }

    /**
     The tracking state of the camera.
     */
    public var trackingState:
ARCamera.TrackingState { get }

    @available(iOS 12.0, *)
    @nonobjc public func unprojectPoint(_
point: CGPoint, ontoPlane planeTransform:
simd_float4x4, orientation:
UIInterfaceOrientation, viewportSize:
```

```
CGSize) -> simd_float3?
}
```