平台 API 文档

基础类：

➢ Future

```
new Li.Future()
```

➢ Variant

```
new Li.Variant()
```

实例方法：

```
toImage() → Image
```

➢ Image

```
new Li.Image()
```

属性：

```
width : Number (readonly)

height : Number (readonly)
```

实例方法：

```
_bits() → Uint8Array
```

```
save(url) → void
```

参数名	类型	描述
url	String	

➢ ImageFuture

```
new Li.ImageFuture()
```

实例方法：

```
_then() → Promise.<Number>
```

➢ ByteArray

```
new Li.ByteArray()
```

实例方法：

```
data() → Uint8Array
```

```
create(byteLength)
```

参数名	类型	描述
byteLength	int	字节长度

静态方法：

```
Li.ByteArray.fromPtr(ptr) → ByteArray
```

参数名	类型	描述

ptr	Number	

➢ ByteArrayFuture

```
new Li.ByteArrayFuture()
```

实例方法：

```
then() → Promise.<Number>
```

➢ Cartographic

属性：

```
longitude : Number (readonly)
latitude : Number (readonly)
height : Number (readonly)
```

实例方法：

```
toDegrees() → Cartographic
```

```
toRadius() → Cartographic
```

```
toCartesian3() → Cartesian3
```

```
toVector3() → Vector3
```

静态方法：

```
Li.Cartographic.fromDegrees(longitude, latitude, height) → Cartographic
```

参数名	类型	描述
longitude	Number	
latitude	Number	
height	Number	

```
Li.Cartographic.fromCartesian(Cartesian3) → Cartographic
```

```
Li.Cartographic.create(longitude, latitude, height) → Cartographic
```

参数名	类型	描述
longitude	Number	
latitude	Number	
height	Number	

➢ Cartesian2

属性：

```
x : Number (readonly)

y : Number (readonly)
```

实例方法：

```
magnitude() → Number
```

```
magnitudeSquared() → Number
```

```
toVector2() → Vector2
```

静态方法：

```
Li.Cartesian2.create(x, y) → Cartesian2
```

参数名	类型	描述
x	Number	
y	Number	

➢ Cartesian3

属性：

```
x : Number (readonly)

y : Number (readonly)

z : Number (readonly)
```

实例方法：

```
normalize() → Cartesian3
```

```
magnitude() → Number
```

```
magnitudeSquared() → Number
```

```
length() → Number
```

```
lengthSquared() → Number
```

```
toVector3() → Vector3
```

```
toCartographic() → Cartographic
```

静态方法：

```
Li.Cartesian3.distance(left, right) → Number
```

参数名	类型	描述
left	Cartesian3	
right	Cartesian3	

```
Li.Cartesian3.dot(left, right) → Number
```

参数名	类型	描述
left	Cartesian3	
right	Cartesian3	

```
Li.Cartesian3.cross(left, right) → Cartesian3
```

参数名	类型	描述
left	Cartesian3	
right	Cartesian3	

```
Li.Cartesian3.lerp(start,end,t) → Cartesian3
```

参数名	类型	描述
start	Cartesian3	
end	Cartesian3	
t	Number	

```
Li.Cartesian3.fromDegrees(longitude,latitude,height) → Cartesian3
```

参数名	类型	描述
longitude	Number	
latitude	Number	
height	Number	

```
Li.Cartesian3.fromRadians(longitude,latitude,height) → Cartesian3
```

参数名	类型	描述
longitude	Number	
latitude	Number	
height	Number	

```
Li.Cartesian3.fromCartographic(Cartographic) → Cartesian3
```

```
Li.Cartesian3.create(x, y, z) → Cartesian3
```

参数名	类型	描述
x	Number	
y	Number	
z	Number	

➢ Cartesian4

属性：

```
x : Number (readonly)

y : Number (readonly)

z : Number (readonly)

w : Number (readonly)
```

实例方法：

```
magnitude() → Number
```

```
magnitudeSquared() → Number
```

```
normalize() → Cartesian4
```

```
toVector4() → Vector4
```

静态方法：

```
Li.Cartesian4.create(x, y, z, w) → Cartesian4
```

参数名	类型	描述
x	Number	
y	Number	
z	Number	
w	Number	

➢ Vector2

属性：

```
x : Number

y : Number
```

实例方法：

```
normalize() → Vector2
```

```
toCartesian2() → Cartesian2
```

静态方法：

```
Li.Vector2.create(x, y) → Vector2
```

参数名	类型	描述

x	Number	
y	Number	

➢ Vector3

属性：

```
x : Number
y : Number
z : Number
```

实例方法：

```
normalize() → Vector3
```

```
magnitude() → Number
```

```
magnitudeSquared() → Number
```

```
length() → Number
```

```
lengthSquared() → Number
```

```
distance(point) → Number
```

参数名	类型	描述
point	Vector3	

```
distanceToPoint(point) → Number
```

参数名	类型	描述

point	Vector3	

minimumComponent() → Number

maximumComponent() → Number

clone(result) → Vector3

参数名	类型	描述
result	Vector3	

toCartesian3() → Cartesian3

静态方法：

Li.Vector3.equals(a, b) → Boolean

参数名	类型	描述
a	Vector3	
b	Vector3	

Li.Vector3.equalsEpsilon(a, b, absoluteEpsilon) → Boolean

参数名	类型	描述
a	Vector3	
b	Vector3	
absoluteEpsilon	Number	

Li.Vector3.dot(v1, v2) → Number

v1	Vector3	
v2	Vector3	

Li.Vector3.cross(v1, v2) → Vector3

参数名	类型	描述
v1	Vector3	
v2	Vector3	

Li.Vector3.min(v1, v2) → Vector3

参数名	类型	描述
v1	Vector3	
v2	Vector3	

Li.Vector3.max(v1, v2) → Vector3

参数名	类型	描述
v1	Vector3	
v2	Vector3	

Li.Vector3.lerp(start,end,t) → Vector3

参数名	类型	描述
start	Vector3	
end	Vector3	
t	Number	

Li.Vector3.abs(Vector3) → Vector3

Li.Vector3.minimumByComponent(a, b) → Vector3

参数名	类型	描述
a	Vector3	
b	Vector3	

Li.Vector3.maximumByComponent(a, b) → Vector3

参数名	类型	描述

a	Vector3	
b	Vector3	

Li.Vector3.negate(Vector3) → Vector3

Li.Vector3.add(a, b) → Vector3

参数名	类型	描述
a	Vector3	
b	Vector3	

Li.Vector3.subtract(a, b) → Vector3

参数名	类型	描述
a	Vector3	
b	Vector3	

Li.Vector3.multiplyComponents(a, b) → Vector3

参数名	类型	描述
a	Vector3	
b	Vector3	

Li.Vector3.multiplyByScalar(a, scalar) → Vector3

参数名	类型	描述
a	Vector3	
scalar	Number	

Li.Vector3.divideComponents(a, b) → Vector3

参数名	类型	描述
a	Vector3	
b	Vector3	

```
Li.Vector3.divideByScalar(a, scalar) → Vector3
```

参数名	类型	描述
a	Vector3	
scalar	Number	

```
Li.Vector3.create(x, y, z) → Vector3
```

参数名	类型	描述
x	Number	
y	Number	
z	Number	

➢ Vector4

属性：

```
x : Number

y : Number

Z : Number

w : Number
```

实例方法：

```
toCartesian4() → Cartesian4
```

静态方法：

```
Li.Vector4.create(x, y, z, w) → Vector4
```

参数名	类型	描述
x	Number	
y	Number	
z	Number	
w	Number	

➢ Quaternion

属性：

```
x : Number

y : Number

Z : Number

w : Number

scalar : Number

vector : Number
```

实例方法：

```
isIdentity() → Boolean
```

```
normalize() → Quaternion
```

```
inverted() → Quaternion
```

```
conjugated() → Quaternion
```

```
rotatedVector(Vector3) → Vector3
```

```
toVector4() → Vector4
```

```
toEulerAngles() → Vector3
```

```
toRotationMatrix() → Matrix3
```

```
getAxes(xAxis, yAxis, zAxis) → void
```

参数名	类型	描述
xAxis	Vector3	
yAxis	Vector3	
zAxis	Vector3	

静态方法：

```
Li.Quaternion.fromAxisAndAngle(xAxis, angle) → Quaternion
```

参数名	类型	描述
xAxis	Vector3	
angle	Number	

```
Li.Quaternion.fromEulerAngles(Vector3) → Quaternion
```

```
Li.Quaternion.fromRotationMatrix(Matrix3) → Quaternion
```

```
Li.Quaternion.fromAxes(xAxis, yAxis, zAxis) → Quaternion
```

参数名	类型	描述
xAxis	Vector3	
yAxis	Vector3	
zAxis	Vector3	

```
Li.Quaternion.fromDirection(direction, up) → Quaternion
```

参数名	类型	描述
direction	Vector3	
up	Vector3	

```
Li.Quaternion.rotationTo(from, to) → Quaternion
```

参数名	类型	描述
from	Vector3	
to	Vector3	

```
Li.Quaternion.slerp(q1, q2, t) → Quaternion
```

参数名	类型	描述
q1	Quaternion	
q2	Quaternion	
t	Number	

```
Li.Quaternion.nlerp(q1, q2, t) → Quaternion
```

参数名	类型	描述
q1	Quaternion	
q2	Quaternion	
t	Number	

```
Li.Quaternion.create(x, y, z, w) → Quaternion
```

参数名	类型	描述
x	Number	
y	Number	
z	Number	
w	Number	

➢ Matrix3

实例方法：

```
row(index) → Vector3
```

参数名	类型	描述
index	Number	

```
setRow(index, value) → void
```

参数名	类型	描述
index	Number	
value	Vector3	

```
column(index) → Vector3
```

参数名	类型	描述
index	Number	

```
setColumn(index, value) → void
```

参数名	类型	描述
index	Number	
value	Vector3	

```
isIdentity() → Boolean
```

```
setToIdentity() → void
```

静态方法：

```
Li.Matrix3.multiply(left, right) → Matrix3
```

参数名	类型	描述
left	Matrix3	
right	Matrix3	

```
Li.Matrix3.multiplyByVector3(matrix, vector) → Vector3
```

参数名	类型	描述
matrix	Matrix3	
vector	Vector3	

```
Li.Matrix3.create() → Matrix3
```

➢ Matrix4

实例方法：

```
row(index) → Vector4
```

参数名	类型	描述
index	Number	

```
setRow(index, value) → void
```

参数名	类型	描述
index	Number	
value	Vector4	

```
column(index) → Vector4
```

参数名	类型	描述
index	Number	

```
setColumn(index, value) → void
```

参数名	类型	描述
index	Number	
value	Vector4	

```
isIdentity() → Boolean
```

```
setToIdentity() → void
```

```
translation() → Vector3
```

```
setTranslation(Vector3) → void
```

```
rotationMatrix() → Vector3
```

```
scale(Vector3) → void
```

```
inverted() → Matrix4
```

```
inverseTransformation() → Matrix4
```

```
transposed() → Matrix4
```

```
compose(position, orientation, scale)  → void
```

参数名	类型	描述
position	Vector3	
orientation	Quaternion	
scale	Vector3	

```
decompose(position, orientation, scale)  → void
```

参数名	类型	描述
position	Vector3	
orientation	Quaternion	
scale	Vector3	

```
optimize() → void
```

```
map(Vector3) → Vector3
```

```
mapVector(Vector3) → Vector3
```

静态方法：

```
Li.Matrix4.multiply(left, right) → Matrix4
```

参数名	类型	描述
left	Matrix4	
right	Matrix4	

```
Li.Matrix4.multiplyByVector3(matrix, vector) → Vector3
```

参数名	类型	描述
matrix	Matrix4	
vector	Vector3	

```
Li.Matrix4.multiplyByVector4(matrix, vector) → Vector4
```

参数名	类型	描述
matrix	Matrix4	
vector	Vector4	

```
Li.Matrix4.fromTranslation(Vector3) → Matrix4
```

```
Li.Matrix4.fromScale(Vector3) → Matrix4
```

```
Li.Matrix4.fromRotationTranslation(rotation, translation) → Matrix4
```

参数名	类型	描述

rotation	Matrix3	
translation	Vector3	

```
Li.Matrix4.fromTranslationRotationScale(translation, rotation,
  scale) → Matrix4
```

参数名	类型	描述
translation	Vector3	
rotation	Matrix3	
scale	Vector3	

```
Li.Matrix4.computeViewportTransformation(x, y, width, height,
nearDepthRange, farDepthRange) → Matrix4
```

参数名	类型	描述
x	Number	
y	Number	
width	Number	
height	Number	
nearDepthRange	Number	
farDepthRange	Number	

```
Li.Matrix4.computePerspectiveOffCenter(left, right, bottom, top,
nearPlane, farPlane) → Matrix4
```

参数名	类型	描述
left	Number	
right	Number	
bottom	Number	
top	Number	
nearPlane	Number	
farPlane	Number	

```
Li.Matrix4.computeInfinitePerspectiveOffCenter(left, right, bottom, top,
nearPlane) → Matrix4
```

参数名	类型	描述

left	Number	
right	Number	
bottom	Number	
top	Number	
nearPlane	Number	
farPlane	Number	

```
Li.Matrix4.create() → Matrix4
```

➢ Rectangle

属性：

```
west : Number (readonly)

south : Number (readonly)

east : Number (readonly)

north : Number  (readonly)
```

实例方法：

```
width() → Number
```

```
height() → Number
```

```
computeWidth() → Number
```

```
computeHeight() → Number
```

```
center() → Rectangle
```

```
southwest() → Rectangle
```

```
northwest() → Rectangle
```

```
northeast() → Rectangle
```

```
southeast() → Rectangle
```

```
combine(Rectangle) → void
```

```
combinePoint(Cartographic) → void
```

```
contains(cartographic) → Boolean
```

```
intersected(Rectangle) → Boolean
```

```
bounds() → BoundingSphere
```

```
toVector4() → Vector4
```

```
toCartesian4() → Cartesian4
```

```
toDegrees() → Rectangle
```

```
toRadians() → Rectangle
```

静态方法：

```
Li.Rectangle.fromDegrees(w, s, e, n) → Rectangle
```

参数名	类型	描述
w	Number	
s	Number	
e	Number	
n	Number	

```
Li.Rectangle.fromRadians(w, s, e, n) → Rectangle
```

参数名	类型	描述
w	Number	
s	Number	
e	Number	
n	Number	

```
Li.Rectangle.intersection(rectangle, otherRectangle) → Rectangle
```

参数名	类型	描述
rectangle	Rectangle	
otherRectangle	Rectangle	

```
Li.Rectangle.simpleIntersection(rectangle, otherRectangle) → Rectangle
```

参数名	类型	描述
rectangle	Rectangle	
otherRectangle	Rectangle	

```
Li.Rectangle.create(w, s, e, n) → Rectangle
```

参数名	类型	描述
w	Number	
s	Number	
e	Number	
n	Number	

➢ BoundingSphere

属性：

```
center : Vector3 (readonly)

radius : Number (readonly)
```

实例方法：

```
contains(BoundingSphere) → Boolean
```

```
merge(BoundingSphere) → void
```

```
expand(Vector3) → void
```

```
transform(Matrix4) → void
```

```
computePlaneDistances(position, direction) → Number
```

参数名	类型	描述
position	Vector3	
direction	Vector3	

```
distanceTo(Vector3) → Number
```

静态方法：

```
Li.BoundingSphere.fromOrientedBoundingBox(OrientedBoundingBox)

 → BoundingSphere
```

```
Li.BoundingSphere.fromAxisAlignedBoundingBox(BoxAxisAlignedBoundingBox)
 → BoundingSphere
```

```
Li.BoundingSphere.create(c, r) → BoundingSphere
```

参数名	类型	描述
c	Vector3	
r	Number	

➢ AxisAlignedBoundingBox

属性：

```
minimum : Vector3 (readonly)
maximum : Vector3 (readonly)
center : Vector3
```

实例方法：

```
extent() →  Vector3
```

```
contains(Vector3) → Boolean
```

```
transform(Matrix4) → void
```

静态方法：

```
Li.AxisAlignedBoundingBox.create(min, max) → AxisAlignedBoundingBox
```

参数名	类型	描述
min	Vector3	
max	Vector3	

➢ OrientedBoundingBox

属性：

```
center : Vector3 (readonly)

halfAxes : Matrix3 (readonly)
```

实例方法：

```
dirU() → Vector3
```

```
dirV() → Vector3
```

```
dirW() → Vector3
```

```
size() → Vector3
```

```
matrix() → Matrix4
```

```
rectangle() → Rectangle
```

```
localAABB() → AxisAlignedBoundingBox
```

```
distanceTo(Vector3) → Number
```

```
distanceSquaredTo(Vector3) → Number
```

静态方法：

```
Li.OrientedBoundingBox.fromRectangle(rectangle, minimumHeight,
  maximumHeight, ellipsoid) → OrientedBoundingBox
```

参数名	类型	描述
rectangle	Rectangle	
minimumHeight	Number	
maximumHeight	Number	
ellipsoid	Ellipsoid	

```
Li.OrientedBoundingBox.fromCartesianAABB(cartesian, box)

→ OrientedBoundingBox
```

参数名	类型	描述
cartesian	Vector3	
box	AxisAlignedBoundingBox	

```
Li.OrientedBoundingBox.create(center, halfAxes) → OrientedBoundingBox
```

参数名	类型	描述
center	Vector3	
halfAxes	Matrix3	

➢ BoundingRegion

属性：

```
rectangle : Rectangle (readonly)

minimumHeight : Number (readonly)

maximumHeight : Number (readonly)
```

实例方法：

```
contains(Cartographic) → Boolean
```

```
combine(BoundingRegion) → void
```

```
distanceTo(a, b) → Number
```

参数名	类型	描述
a	Cartesian3	
b	Cartographic	

静态方法：

```
Li.BoundingRegion.fromOrientedBoundingBox(OrientedBoundingBox)
→ BoundingRegion
```

```
Li.BoundingRegion.create(region, minHeight, maxHeight) → BoundingRegion
```

参数名	类型	描述
region	Rectangle	
minHeight	Number	
maxHeight	Number	

➢ BoundingVolume

属性：

```
center : Vector3 (readonly)
boundingSphere : BoundingSphere (readonly)
orientedBoundingBox : OrientedBoundingBox (readonly)
```

实例方法：

```
intersectPlane(Plane) → Intersect
```

```
computeVisibility(CullingVolume) → Intersect
```

```
distanceTo(a, b) → Number
```

参数名	类型	描述
a	Cartesian3	
b	Cartographic	

静态方法：

```
Li.BoundingVolume.fromBoundingSphere(BoundingSphere)
→ BoundingVolume
```

```
Li.BoundingVolume.fromOrientedBoundingBox(OrientedBoundingBox)
→ BoundingVolume
```

```
Li.fromBoundingRegion.fromOrientedBoundingBox(BoundingRegion)
→ BoundingVolume
```

➢ Ray

属性：

```
origin : Vector3 (readonly)
direction : Vector3 (readonly)
```

静态方法：

```
Li.Ray.create(origin, direction) → Ray
```

参数名	类型	描述
origin	Vector3	
direction	Vector3	

➢ Plane

属性：

```
normal : Vector3 (readonly)

distance : Number (readonly)
```

静态方法：

```
Li.Plane.create(n, d)
```

参数名	类型	描述
n	Vector3	
d	Number	

➢ Color

```
new Li.Color()
```

```
new Li.Color(r, g, b, a)
```

参数名	类型	描述
r	Number	
g	Number	
b	Number	
a	Number	

属性：

```
red : Number
green : Number
blue : Number
alpha : Number
redF : Number
greenF : Number
blueF : Number
alphaF : Number
r : Number
g : Number
b : Number
a : Number
```

静态方法：

```
Li.Color.fromRgb(r, g, b, a) → Color
```

参数名	类型	描述
r	Number	
g	Number	
b	Number	
a	Number	

```
Li.Color.fromRgbF(r, g, b, a) → Color
```

参数名	类型	描述
r	Number	
g	Number	
b	Number	
a	Number	

➢ Buffer

静态方法：

```
Li.Buffer.createVertexBuffer(ba, stride) → Buffer
```

参数名	类型	描述
ba	TypedArray	
stride	Number	

```
Li.Buffer.createIndexBuffer(ba, stride) → Buffer
```

参数名	类型	描述
ba	TypedArray	
stride	Number	

➢ GeometryAttribute

静态方法：

```
Li.GeometryAttribute.create(buffer, offsetBytes, components,
  componentDataType) → GeometryAttribute
```

参数名	类型	描述
buffer	Buffer	
offsetBytes	Number	
components	Number	
componentDataType	Number	

```
Li.GeometryAttribute.createPositionAttribute(buffer, offsetBytes,
  components) → GeometryAttribute
```

参数名	类型	描述
buffer	Buffer	
offsetBytes	Number	
components	Number	

```
Li.GeometryAttribute.createNormalAttribute(buffer, offsetBytes,
  components) → GeometryAttribute
```

参数名	类型	描述
buffer	Buffer	
offsetBytes	Number	
components	Number	

```
Li.GeometryAttribute.createTangentAttribute(buffer, offsetBytes,
  components) → GeometryAttribute
```

参数名	类型	描述
buffer	Buffer	
offsetBytes	Number	
components	Number	

```
Li.GeometryAttribute.createTexCoordAttribute(buffer, offsetBytes,
  components) → GeometryAttribute
```

参数名	类型	描述
buffer	Buffer	
offsetBytes	Number	
components	Number	

➢ Geometry

```
new Li.Geometry()
```

属性：

```
indexBuffer : Buffer (readonly)
```

实例方法：

```
addAttribute(buffer, offset, components, attrType)
```

参数名	类型	描述
buffer	Buffer	
offset	Number	
components	Number	
attrType	Number	

➢ TextureImage

```
new Li.TextureImage()
```

属性：

```
width : Number

height : Number

depth : Number

layers : Number

mipLevels : Number

textureFormat : TextureFormat
```

```
pixelFormat : TextureFormat
```

```
pixelType : PixelType
```

实例方法：

```
loadImage(url) → void
```

参数名	类型	描述
url	String	

```
setData(data, blockSize, isCompressed) → void
```

参数名	类型	描述
data	Buffer	
blockSize	Number	
isCompressed	Boolean	

➢ Texture

```
new Li.Texture()
```

属性：

```
width : Number

height : Number

depth : Number

layers : Number

target : Target (onlyread)

format : TextureFormat (onlyread)
```

实例方法：

```
addTextureImage(TextureImage) → void
```

➢ TextureCubeMap

```
new Li.TextureCubeMap()
```

➢ Texture2DArray

```
new Li.Texture2DArray()
```

➢ Texture3D

```
new Li.Texture3D()
```

➢ Material

```
new Li.Material()
```

属性：

```
shadingModel : ShadingModel

color : Color

opacity : Number

bothSided : Boolean
```

```
shaderProgram : ShaderProgram

texture : Texture

roughness : Number

metallic : Number
```

```
reflectance : Number
```

➢ Component

属性：

```
entity : Entity  (readonly)
transform : Transform  (readonly)
renderer : GeometryRenderer  (readonly)
camera : Camera  (readonly)
light : Light  (readonly)
rigidBody : RigidBody  (readonly)
skin : Skin  (readonly)
animator : Animator  (readonly)
```

实例方法：

```
addComponent(Component) → void
```

➢ Transform

属性：

```
cartographic : Cartographic
matrix : Matrix4
worldMatrix : Matrix4
translation : Vector3
position : Vector3
rotation : Quaternion
```

```
rotationX : Number

rotationY : Number

rotationZ : Number

eulerAngles : Vector3

xaxis : Vector3 (readonly)

yaxis : Vector3 (readonly)

zaxis : Vector3 (readonly)

right : Vector3 (readonly)

forward : Vector3 (readonly)

up : Vector3 (readonly)

scale : Number

scale3D : Vector3
```

实例方法：

```
lookAt(Vector3) → void
```

```
setTransform(translation, rotation, scale) → void
```

参数名	类型	描述
translation	Vector3	
rotation	Quaternion	
scale	Vector3	

```
setAxes(xAxis, yAxis, zAxis) → void
```

参数名	类型	描述
xAxis	Vector3	
yAxis	Vector3	
zAxis	Vector3	

```
localToWorldMatrix() → Matrix4
```

```
worldToLocalMatrix() → Matrix4
```

➢ Model

```
new Li.Model(url)
```

```
new Li.Model(url, offset, scale, rotation)
```

参数名	类型	描述
url	String	
offset	Vector3	
scale	Vector3	
rotation	Quaternion	

属性：

```
url : String

offset : Vector3

scale : Vector3

rotation : Quaternion
```

➢ Tileset

```
new Li.Tileset(url)
```

参数名	类型	描述
url	String	

属性：

```
url : String (readonly)

rectangle : Rectangle

streamingMode : Boolean

skipLevelOfDetail : Boolean
```

```
clipLevelOfDetail : Boolean

geometricErrorScale : Number

genMeshNormals : Boolean
```

实例方法：

```
addFlattenMask(FlattenMask) → void
```

```
removeFlattenMask(FlattenMask) → void
```

```
setHeader(FlattenMask) → void
```

➢ Entity

```
new Li.Entity()
```

属性：

```
tag : String

parentEntity : Entity (readonly)

childEntities : EntityList (readonly)

transform : Transform (readonly)

renderer : GeometryRenderer (readonly)
```

```
camera : Camera (readonly)

light : Light (readonly)

rigidBody : RigidBody (readonly)

skin : Skin (readonly)

animator : Animator (readonly)
```

实例方法：

```
addComponent(Component) → void
```

静态方法：

```
Li.Entity.root() → Entity
```

➢ GeometryRenderer

```
new Li.GeometryRenderer()
```

属性：

```
type : Type
primitiveType : PrimitiveType
instanceCount : Number
primitiveCount : Number
indexOffset : Number
firstInstance : Number
firstVertex : Number
castShadow : Boolean
receiveShadow : Boolean
```

```
boundingVolume : BoundingVolume (readonly)
```

```
instanceBuffer : Buffer
```

```
geometry : Geometry
```

```
material : Material
```

➢ FrameAction

```
new Li.FrameAction()
```

➢ Camera

```
new Li.Camera()
```

属性：

```
projectionType : ProjectionType

aperture : Number

shutterSpeed : Number

sensitivity : Number

nearPlane : Number

farPlane : Number

fov : Number

fovy : Number

fieldOfView : Number
```

```
aspectRatio : Number

left : Number

right : Number

bottom : Number

top : Number
```

实例方法：

```
viewMatrix() → Matrix4
```

```
projectionMatrix() → Matrix4
```

```
viewportOrthographic() → Matrix4
```

```
worldToCameraMatrix() → Matrix4
```

```
cameraToWorldMatrix() →  Matrix4
```

```
getPixelSize(boundingSphere, drawingBufferWidth, drawingBufferHeight)
  →  Number
```

参数名	类型	描述
boundingSphere	BoundingSphere	
drawingBufferWidth	Number	
drawingBufferHeight	Number	

```
getPixelDimensions(drawingBufferWidth, drawingBufferHeight, distance)
  →  Cartesian2
```

参数名	类型	描述
drawingBufferWidth	Number	
drawingBufferHeight	Number	
distance	Number	

```
computeViewRectangle() →  Rectangle
```

```
screenPointToRay(x, y)    →  Ray
```

参数名	类型	描述

x	Number	
y	Number	

```
screenToWorldPoint(Vecotor3) → Vecotor3
```

```
worldToScreenPoint(Vecotor3) → Vecotor3
```

```
flyTo(Cartographic) → void
```

```
cameraController() → CameraController
```

➢ CameraController

属性：

```
enableInputs : Boolean

enableZoom : Boolean

enableRotate : Boolean

enableTilt : Boolean

enableLook : Boolean

enablePan : Boolean

enableUnderGround : Boolean

minimumCollisionTerrainHeight : Number

positionWC : Vector3 (readonly)
```

```
rightWC : Vector3 (readonly)

directionWC : Vector3 (readonly)

upWC : Vector3 (readonly)

positionCartographic : Cartographic (readonly)

heading : Number (readonly)
```

```
pitch : Number (readonly)

roll : Number (readonly)
```

实例方法：

```
getPickRay(x, y) → Ray
```

参数名	类型	描述
x	Number	
y	Number	

```
setView(destination, heading, pitch, roll) → void
```

参数名	类型	描述
destination	Cartesian3	
heading	Number	
pitch	Number	
roll	Number	

```
rotate(axis, angle) → void
```

参数名	类型	描述
axis	Vector3	
angle	Number	

```
rotateUp(angle) → void
```

参数名	类型	描述
angle	Number	

```
rotateDown(angle) → void
```

参数名	类型	描述
angle	Number	

```
rotateRight(angle) → void
```

参数名	类型	描述
angle	Number	

`rotateLeft(angle)` → `void`

参数名	类型	描述
angle	Number	

`move(dir, amount)` → `void`

参数名	类型	描述
dir	Vector3	
amount	Number	

`moveForward(amount)` → `void`

参数名	类型	描述
amount	Number	

`moveBackward(amount)` → `void`

参数名	类型	描述
amount	Number	

`moveUp(amount)` → `void`

参数名	类型	描述
amount	Number	

`moveDown(amount)` → `void`

参数名	类型	描述
amount	Number	

`moveRight(amount)` → `void`

参数名	类型	描述
amount	Number	

moveLeft(amount) → void

参数名	类型	描述
amount	Number	

look(axis, amount) → void

参数名	类型	描述
axis	Vector3	
amount	Number	

lookUp(amount) → void

参数名	类型	描述
amount	Number	

lookDown(amount) → void

参数名	类型	描述
amount	Number	

lookRight(amount) → void

参数名	类型	描述
amount	Number	

lookLeft(amount) → void

参数名	类型	描述
amount	Number	

zoomIn(amount) → void

参数名	类型	描述
amount	Number	

flyTo(destination, duration, heading, pitch, roll) → void

参数名	类型	描述
destination	Vector3	
duration	Number	
heading	Number	
pitch	Number	
roll	Number	

flyToCartographic(destination, duration, heading, pitch, roll) → void

参数名	类型	描述
destination	Cartographic	
duration	Number	
heading	Number	
pitch	Number	
roll	Number	

flyToRectangle(destination, duration, heading, pitch, roll) → void

参数名	类型	描述
destination	Rectangle	
duration	Number	
heading	Number	
pitch	Number	
roll	Number	

pickGlobe(x, y) → Cartesian3

参数名	类型	描述
x	Number	
y	Number	

➢ TextureProjection

```
new Li.TextureProjection()
```

属性：

```
texture : Texture

projectionType : ProjectionType

blendMode : BlendMode

sceneMode : SceneMode

textureFlip : Boolean

showFrustum : Boolean

nearPlane : Number

farPlane : Number

width : Number

height : Number

fov : Number

aspect : Number
```

➢ Viewshed

```
new Li.Viewshed()
```

属性：

```
start : Vector3

end : Vector3

radius : Number
```

```
horizontalAngle : Number

visibleColor : Color

invisibleColor : Color

showDebugFrame : Boolean
```

➢ ClipVolume

```
new Li.ClipVolume()
```

属性：

```
shape : Shape

clipFlag : ClipFlag

size : Vector3

showDebugVolume : Boolean
```

➢ FlattenMask

```
new Li.FlattenMask()
```

属性：

```
maskHeight : Number
```

实例方法：

```
setPoints(Vector3)    →  void
```

➢ RaycastHit

```
new Li.RaycastHit()
```

属性：

```
distance : Number （readonly）
point : Vector3 （readonly）
normal : Vector3 （readonly）
material : Material （readonly）
entity : Entity （readonly）
```

➢ Atmosphere

属性：

```
show : Boolean
intensity : Number
inscatterAmount : Number
cloudOpacity :  Number
cloudIntensity : Number
```

➢ Sun

属性：

```
color : Color
intensity : Number
```

```
haloSize : Number
castShadow :  Number
shadowBias : Number
```

➢ IndirectLight

属性：

```
intensity : Number
diffuseIntensity : Number
specularIntensity :  Number
rotation : Number
```

➢ PostRendering

属性：

```
saturation : Number
contrast : Number
sharpen :  Number
exposure : Number
autoExposure : Boolean
brightThreshold : Number
bloomValue :  Number
```

➢ RenderSystem

实例方法：

```
colorTexture() → Texture
```

```
depthTexture() → Texture
```

```
normalTexture() → Texture
```

```
postRendering() → PostRendering
```

```
waterParameters() → WaterParameters
```

```
renderToImage(width, height) → Image
```

参数名	类型	描述
width	Number	
height	Number	

➢ Ellipsoid

属性：

```
radii : Cartesian3 (readonly)

radiiSquared : Cartesian3 (readonly)

radiiToTheFourth : Cartesian3 (readonly)

oneOverRadii : Cartesian3 (readonly)

oneOverRadiiSquared : Cartesian3 (readonly)

minimumRadius : Number (readonly)

maximumRadius :  Number (readonly)
```

实例方法：

```
cartographicToCartesian(Cartographic) → Cartesian3
```

```
cartesianToCartographic(Cartesian3) → Cartographic
```

```
eastNorthUpToFixedFrame(Cartesian3) → Matrix4
```

```
geodeticSurfaceNormalCartographic(Cartographic) → Cartesian3
```

```
transformPositionToScaledSpace(Cartesian3) → Cartesian3
```

```
scaleToGeocentricSurface(Cartesian3) → Cartesian3
```

```
getSurfaceNormalIntersectionWithZAxis(position, buffer) → Cartesian3
```

参数名	类型	描述
position	Cartesian3	
buffer	Number	

```
geodeticSurfaceNormal(Cartesian3) → Cartesian3
```

```
scaleToGeodeticSurface(cartesian, oneOverRadii, oneOverRadiiSquared,
  centerToleranceSquared) → Cartesian3
```

参数名	类型	描述
cartesian	Cartesian3	
oneOverRadii	Cartesian3	
oneOverRadiiSquared	Cartesian3	
centerToleranceSquared	Number	

静态方法：

```
Li.Ellipsoid.WGS84() → Ellipsoid
```

➢ MapProjection

实例方法：

```
project(Cartographic) → Cartesian3
```

```
unproject(Cartesian3) → Cartographic
```

➢ GeographicProjection

```
new Li.GeographicProjection(Ellipsoid)
```

➢ WebMercatorProjection

```
new Li.WebMercatorProjection(Ellipsoid)
```

➢ GeographicTilingScheme

```
new Li.GeographicTilingScheme(rectangle, numberOfLevelZeroTilesX,
  numberOfLevelZeroTilesY, ellipsoid)
```

参数名	类型	描述
cartesian	Rectangle	

numberOfLevelZeroTilesX	Number	
numberOfLevelZeroTilesY	Number	
ellipsoid	Ellipsoid	

➢ WebMercatorTilingScheme

```
new Li.WebMercatorTilingScheme(numX, numY, ellipsoid)
```

参数名	类型	描述
numX	Number	
numY	Number	
ellipsoid	Ellipsoid	

➢ UrlTemplateImageryProvider

```
new Li.UrlTemplateImageryProvider(url, useWebMercator, maximumLevel,

minimumLevel, tileWidth, tileHeight, hasAlphaChannel)
```

参数名	类型	描述
url	String	
useWebMercator	Boolean	
maximumLevel	Number	
minimumLevel	Number	
tileWidth	Number	
tileHeight	Number	
hasAlphaChannel	Boolean	

➢ ImageryLayer

```
new Li.ImageryLayer(provider, rectangle)
```

参数名	类型	描述

provider	ImageryProvider	
rectangle	Rectangle	

属性：

```
show : Boolean

brightness : Number

contrast : Number

saturation : Number

hue : Number

gamma : Number

alpha :  Number
```

➢ Globe

属性：

```
show : Boolean

baseColor : Color

lightingEnabled : Boolean

ellipsoid : Ellipsoid (readonly)
```

实例方法：

```
setTerrainProviderUrl(url) → void
```

参数名	类型	描述
url	String	

```
setDefaultTerrain() → void
```

```
addImageryLayer(ImageryLayer) → void
```

```
addArcGisMapServerImageryLayer(url) → void
```

参数名	类型	描述
url	String	

```
getHeight(cartographic, includeTerrainSurface) → Number
```

参数名	类型	描述
cartographic	Cartographic	
includeTerrainSurface	Boolean	

```
getHeight(ray, result) → Boolean
```

参数名	类型	描述
ray	Ray	
result	Cartesian3	

```
addFlattenMask(FlattenMask) → void
```

```
removeFlattenMask(FlattenMask) → void
```

➢ Scene

属性：

```
canvas : Canvas (readonly)

rootEntity :  Entity (readonly)

mainCamera : Camera (readonly)
```

```
skybox : Skybox (readonly)

sun : Sun (readonly)

indirectLight : IndirectLight (readonly)

atmosphere : Atmosphere (readonly)

globe : Globe (readonly)

fog : Fog (readonly)
```

实例方法：

```
addEntity(Entity) → void
```

```
raycast(ray, hit) → Boolean
```

参数名	类型	描述
ray	Ray	
hit	RaycastHit	

```
getWorldPositionByMouse() → Vector3
```

```
getFeatureByMouse() → Feature
```

```
selectedFeature() → Feature
```

```
setSelectedFeature(Feature) → void
```

```
featureSelectedColor() → Color
```

```
setFeatureSelectedColor(Color) → void
```

➢ TimeSystem

属性：

```
timeSeconds : Number (readonly)

deltaSeconds : Number (readonly)
```

实例方法：

```
setYear(year) → void
```

参数名	类型	描述
year	Number	

```
setMonth(month) → void
```

参数名	类型	描述
month	Number	

```
setDay(day) → void
```

参数名	类型	描述
day	Number	

```
setHour(hour) → void
```

参数名	类型	描述
hour	Number	

```
setMinute(minute) → void
```

参数名	类型	描述
minute	Number	

```
setSecond(second) → void
```

参数名	类型	描述
second	Number	

➢ Viewer

属性：

```
canvas : Canvas (readonly)

scene :  Scene (readonly)

timeSystem : TimeSystem (readonly)

inputSystem : InputSystem (readonly)

renderSystem :  RenderSystem (readonly)
```

实例方法：

```
addEventListener(type, listener) → void
```

参数名	类型	描述
type	String	
listener	Function	

```
loadQmlFile(url) → void
```

参数名	类型	描述
url	String	

➢ ShapefileLayer

```
new Li.ShapefileLayer()
```

属性：

```
ready : Boolean (readonly)

color :  Color

splitByTiles : Boolean

url : String

srs : String

proj4 : String
```

实例方法：

```
componentComplete() → void
```

➢ RenderLayer

```
new Li.RenderLayer()
```

属性：

```
color :  Color

renderStyle :  RenderStyle
```

➢ ProjectionNode

```
new Li.ProjectionNode()
```

实例方法：

```
setColor(Color) → void
```

```
setSelectedColor(Color) → void
```

```
setStrokeColor(Color) → void
```

```
setStrokeWidth(width) → void
```

参数名	类型	描述
width	String	

```
addPoint(Cartographic) → void
```

```
setPoints(CartographicVector) → void
```

```
setPoint(a, b) → void
```

参数名	类型	描述
a	Cartographic	
b	Cartographic	

```
removePoint(index) → void
```

参数名	类型	描述
index	Number	

➢ ProjectionDataSource

```
new Li.ProjectionDataSource()
```

实例方法：

```
setColor(Color) → void
```

```
addField(field) → void
```

参数名	类型	描述
field	String	

```
setSelectedColor(Color) → void
```

```
setStrokeColor(Color) → void
```

```
setStrokeWidth(width) → void
```

参数名	类型	描述
width	String	

```
loadGeoJson(url) → void
```

参数名	类型	描述
url	String	

➢ ProjectionLayer

```
new Li.ProjectionLayer()
```

属性:

```
textureSize : Number
```

实例方法:

```
addSourceLayer(ShapefileLayer) → void
```

```
addDataSource(ProjectionDataSource) → void
```

```
addNode(ProjectionNode) → void
```

```
moveUp(ProjectionNode) → void
```

```
moveDown(ProjectionNode) → void
```

```
moveToTop(ProjectionNode) → void
```

```
moveToBottom(ProjectionNode) → void
```

```
getFeatureByMouse() → Feature
```

静态方法：

```
Li.ProjectionLayer.globalInstance() → ProjectionLayer
```

➢ WaterLayer

```
new Li.WaterLayer()
```

实例方法：

```
changStyle() → void
```

```
componentComplete() → void
```

➢ ModelLayer

```
new Li.ModelLayer()
```

属性:

```
url : String

cartographic : Cartographic

offset : Vector3

rotation : Vector3

scale : Vector3
```

实例方法:

```
componentComplete() → void
```

```
entity() → void
```

➢ MeshLayer

```
new Li.MeshLayer()
```

属性:

```
url : String

offset : Vector3

rotation : Vector3

scale : Vector3
```

实例方法:

```
instances() → void
```

```
setInstances(ShapefileLayer) → void
```

```
extent() → Rectangle
```

```
texture() → Texture
```

```
componentComplete() → void
```

➢ DecalLayer

```
new Li.DecalLayer()
```

实例方法：

```
addSourceLayer(MeshLayer) → void
```

```
removeSourceLayer(MeshLayer) → void
```

```
clearSourceLayers() → void
```

```
changStyle() → void
```

```
componentComplete() → void
```

➢ TilesetLayer

```
new Li.TilesetLayer()
```

属性:

```
url : String

skipLevelOfDetail : Boolean

clipLevelOfDetail : Boolean

genMeshNormals : Boolean
```

实例方法:

```
tileset() → TilesetLayer
```

```
componentComplete() → void
```

➢ ImageryWrapper

```
new Li.ImageryWrapper()
```

属性:

```
url : String

rectangle : Rectangle

useWebMercator : Boolean

tileWidth : Number

tileHeight : Number

minimumLevel : Number
```

```
hasAlphaChannel : Boolean

useWebMercator : Boolean

isLabel : Boolean
```

➢ TiandituImageryLayer

```
new Li.TiandituImageryLayer()
```

属性:

```
url : String

rectangle : Rectangle

useWebMercator : Boolean

tileWidth : Number

tileHeight : Number

minimumLevel : Number

hasAlphaChannel : Boolean

useWebMercator : Boolean

isLabel : Boolean
```

实例方法:

```
componentComplete() → void
```

➢ Feature

```
new Li.Feature()
```

实例方法:

```
propertyNames() → RegistryKeys
```

```
getProperty(feild) → String
```

参数名	类型	描述
feild	String	

➢ GeoJsonModel

```
new Li.GeoJsonModel()
```

属性：

```
show : Boolean

height : Number

heightField : String

iconUrl : String

selectedIconUrl : String

iconSize : Vector2

iconOffset : Vector2

fontSize : Number

highlight : Boolean

labelField : String

labelOffset : Vector2
```

实例方法：

```
addField(feild) → String
```

参数名	类型	描述
feild	String	

```
setBase3DTileset(Tileset) → void
```

```
addField(feild) → String
```

参数名	类型	描述
feild	String	

```
load(url) → void
```

参数名	类型	描述
url	String	

```
getFeatureCount() → Number
```

```
getProperty(feature, feild) → String
```

参数名	类型	描述
feature	Feature	
feild	String	

静态方法：

```
Li.GeoJsonModel.getSelectedFeature() → GeoJsonModel
```

➢ GeoJsonBillboards

```
new Li.GeoJsonBillboards()
```

属性：

```
iconTextOffset : Vector2

iconTextFontSize : Number
```

```
iconTextColor : Color

iconTextField : String

iconTextPrefix : String

iconTextSuffix : String
```

实例方法：

```
setJson() → void
```

```
setIconEnabled() → void
```

➢ ShapeFeature

```
new Li.ShapeFeature()
```

➢ GeoJsonFeatures

```
new Li.GeoJsonFeatures()
```

实例方法：

```
setJson(json) → void
```

参数名	类型	描述
string	String	

```
setKeyField(newKeyField) → void
```

参数名	类型	描述
newKeyField	String	

```
selectFeatures(bounds, highlight) → RegistryKeys
```

参数名	类型	描述
bounds	PointVector	
highlight	Boolean	

```
selectFeature(ShapeFeature) → void
```

```
deselectFeature(ShapeFeature) → void
```

```
isSelected(ShapeFeature) → boolean
```

```
clearSelection() → void
```

```
addField(feild) → String
```

参数名	类型	描述
feild	String	

```
load(url) → void
```

参数名	类型	描述
url	String	

```
pickFeatureByMouse() → void
```

```
highlightFeature() → void
```

➢ TrackPathRenderer

```
new Li.TrackPathRenderer()
```

实例方法：

```
addPoint(pos, text) → void
```

参数名	类型	描述
pos	Cartographic	
text	String	

```
setWidth(newWidth) → void
```

参数名	类型	描述
newWidth	string	

```
setColor(Color) → void
```

```
setIconUrl(newIconUrl) → void
```

参数名	类型	描述
newIconUrl	string	

```
setSelectedIconUrl(Url) → void
```

参数名	类型	描述
Url	string	

```
setIconSize(Vector2) → void
```

```
setIconOffset(Vector2) → void
```

```
setLabelColor(Color) → void
```

```
setFontSize(newFontSize) → void
```

参数名	类型	描述
newFontSize	Number	

```
setArrowIconUrl(url) → void
```

参数名	类型	描述
url	string	

```
setArrowIconSize(Vector2) → void
```

```
setArrowIconRotation(rotation) → void
```

参数名	类型	描述
rotation	string	

```
getSelectedByMouse() → Number
```

```
getScreenPosition() → Vector2
```

➢ FontManager

静态方法：

```
Li.FontManager.setFont(fontType) → void
```

参数名	类型	描述
fontType	String	

```
Li.FontManager.fontTitles(fontType) → RegistryKeys
```

绘图类：

➢ DoubleFuture

```
new Li.DoubleFuture()
```

实例方法：

```
then() → Promise.<Number>
```

➢ StringFuture

```
new Li.StringFuture()
```

实例方法：

```
then() → Promise.<String>
```

➢ FileSystem

静态方法：

```
Li.FileSystem.setGlobalFont(fontRccUrl) → void
```

参数名	类型	描述
fontRccUrl	String	

```
Li.FileSystem.setFont(fontName) → boolean
```

参数名	类型	描述
fontName	String	

```
Li.FileSystem.qtReadFiles(event) → void
```

参数名	类型	描述
event	Object	

```
Li.FileSystem.qtReadFileContent(event) → void
```

参数名	类型	描述
event	Object	

```
Li.FileSystem.saveFile(data, length, fileNameHint) → void
```

参数名	类型	描述
data	String	
length	Number	
fileNameHint	String	

```
Li.FileSystem.saveRenderImage(width, height, name, format) → void
```

参数名	类型	描述
width	Number	
height	Number	
name	String	
format	String	

```
Li.FileSystem.print() → void
```

➢ Billboard

```
new Li.Billboard()
```

属性：

```
url : String

width : Number

height : Number

scale : Number

rotation : Number

translate : Cartesian2

position : Cartesian3

color : Color

verticalOrigin : Vertical

horizontalOrigin : Horizontal
```

实例方法：

```
translucencyByDistance() → Cartesian4
```

```
setTranslucencyByDistance(Cartesian4) → void
```

```
distanceDisplayCondition() → Cartesian2
```

```
setDistanceDisplayCondition(Cartesian2) → void
```

```
scaleByDistance() → Cartesian4
```

```
setScaleByDistance(Cartesian4) → void
```

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

➢ BillboardEntity

```
new Li.BillboardEntity()
```

属性:

```
url : String

imageWidth : Number

imageHeight : Number

scale : Number

rotation : Number

translate : Cartesian2

position : Vector3

vertical : Vertical

horizontal : Horizontal

altitude : Number

altitudeMethod : AltitudeMethod
```

实例方法:

```
collection() → BillboardCollection
```

```
setCollection(BillboardCollection) → void
```

➢ VisualEntity

```
new Li.VisualEntity()
```

实例方法：

```
dirty() → Boolean
```

```
setDirty(dirty) → void
```

参数名	类型	描述
dirty	Boolean	

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
scaleByDistance() → Cartesian4
```

```
setScaleByDistance(Cartesian4) → void
```

```
translucencyByDistance() → Cartesian4
```

```
setTranslucencyByDistance(Cartesian4) → void
```

➢ Label3D

```
new Li.Label3D()
```

属性:

```
position : Vector3
background : Color
strokeColor : Color
fontColor : Color
fontSize : Number
fontBold : Boolean
fontItalic : Boolean
fontUnderline : Boolean
imageWidth : Number
imageHeight : Number
labelWidth : Number (readonly)
labelHeight : Number (readonly)
mix : Boolean (readonly)
vertical : Vertical
horizontal : Horizontal
textVertical : Vertical
textHorizontal : Horizontal
frameWidth : Number
scale : Number
```

```
frameUrl : String

font : Number

text : String

url : String
```

实例方法：

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
collection() → BillboardCollection
```

```
setCollection(BillboardCollection) → void
```

```
iconBB() → Billboard
```

```
textBB() → Billboard
```

➢ LabelProjection

```
new Li.LabelProjection()
```

属性：

```
position : Vector3
background : Color
fontColor : Color
text : String
font : String
fontSize : Number
fontBold : Boolean
fontItalic : Boolean
fontUnderline : Boolean
width : Number
height : Number
pixelWidth : Number (readonly)
pixelHeight : Number (readonly)
frameUrl : String
frameWidth : Number
rotation : Number
```

➢ BillboardCollection

```
new Li.BillboardCollection()
```

实例方法：

```
add(Billboard) → void
```

```
remove(Billboard) → Boolean
```

```
removeAll() → void
```

➢ Line

```
new Li.Line()
```

属性：

```
color : Color

alpha : Number
```

实例方法：

```
add(Billboard) → void
```

```
remove(Billboard) → Boolean
```

```
removeAll() → void
```

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
width() → Number
```

```
setWidth(width) → void
```

参数名	类型	描述
width	Number	

```
addPoint(Vector3) → void
```

```
removeLast() → void
```

```
draw() → void
```

```
redraw() → void
```

```
lineTo(Vector3) → void
```

```
end() → void
```

```
modify() → void
```

```
numOfPoints() → Number
```

➢ Polyline3D

```
new Li.Polyline3D()
```

属性：

```
color : Color

alpha : Number
```

实例方法：

```
glowMaterial() → Boolean
```

```
setGlowMaterial(glowMaterial) → Boolean
```

参数名	类型	描述
glowMaterial	Boolean	

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
width() → Number
```

```
setWidth(width) → void
```

参数名	类型	描述
width	Number	

```
addPoint(Vector3) → void
```

```
lineTo(Vector3) → void
```

```
removeLast() → void
```

```
draw() → void
```

```
end() → void
```

```
modify() → void
```

```
numOfPoints() → Number
```

➢ Polygon3D

```
new Li.Polygon3D()
```

属性：

```
color : Color
alpha : Number
fillAlpha : Number
brushStyle : BrushStyle
image : String
```

实例方法：

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
fillColor() → Color
```

```
setFillColor(color, style) → void
```

参数名	类型	描述
color	Color	
style	BrushStyle	

```
width() → Number
```

```
setWidth(width) → void
```

参数名	类型	描述
width	Number	

```
add(Vector3) → void
```

```
lineTo(Vector3) → void
```

```
removeLast() → void
```

```
draw() → void
```

```
end() → void
```

```
modify() → void
```

```
numOfPoints() → Number
```

➢ ArcLayer

```
new Li.ArcLayer()
```

属性：

```
lineWidth : Number

height : Number

animationTimer : Number

animationRun : Boolean
```

实例方法：

```
addString(str) → void
```

参数名	类型	描述
str	String	

```
ndjsonFile() → String
```

```
setNDJsonFile(url) → void
```

参数名	类型	描述
url	String	

```
startField() → String
```

```
setStartField(url) → void
```

参数名	类型	描述
url	String	

```
endField() → String
```

```
setEndField(field) → void
```

参数名	类型	描述
field	String	

```
nameField() → String
```

```
setNameField(field) → void
```

参数名	类型	描述
field	String	

```
startColor() → Color
```

```
setStartColor(Color) → void
```

```
endColor() → Color
```

```
setEndColor(Color) → void
```

```
create() → Boolean
```

➢ GeoJsonLayer

```
new Li.GeoJsonLayer()
```

属性：

```
geoJson : String
```

➢ HexagonLayer

```
new Li.HexagonLayer()
```

属性：

```
radius : String

color : Color
```

实例方法：

```
create() → Boolean
```

➢ ScatterplotLayer

```
new Li.ScatterplotLayer()
```

属性：

```
radius : String

strokeColor : Color
```

```
fillColor : Color
```

实例方法：

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
scaleByDistance() → Vector4
```

```
setScaleByDistance(Vector4) → void
```

```
translucencyByDistance() → Vector4
```

```
setTranslucencyByDistance(Vector4) → void
```

```
create() → Boolean
```

➢ GeoJsonLabelLayer

```
new Li.GeoJsonLabelLayer()
```

属性：

```
url : String

background : Color

strokeColor : Color

fontColor : Color

font : String

fontSize : Number

fontBold : Boolean

fontItalic : Boolean

fontUnderline : Boolean

scale : Number

imageWidth : Number

imageHeight : Number

mix : Boolean

vertical : Vertical

horizontal : Horizontal

textVertical : Vertical

textHorizontal : Horizontal
```

实例方法：

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

nameField() → String

setNameField(field) → void

参数名	类型	描述
field	String	

terrainField() → String

setTerrainField(field) → void

参数名	类型	描述
field	String	

setFieldsDDC() → DistanceDisplayField

setCategory(CategoryField) → void

setCategoryEnabled(field, enabled) → void

参数名	类型	描述
field	String	
enabled	Boolean	

scaleByDistance() → Vector4

setScaleByDistance(Vector4) → void

"translucencyByDistance() → Vector4

setTranslucencyByDistance(Vector4) → void

```
create() → Boolean
```

➢ GeoJsonPointLayer

```
new Li.GeoJsonPointLayer()
```

属性:

```
url : String

scale : Number

width : Number

height : Number

vertical : Vertical

horizontal : Horizontal
```

实例方法:

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
scaleByDistance() → Vector4
```

```
setScaleByDistance(Vector4) → void
```

```
"translucencyByDistance() → Vector4
```

```
setTranslucencyByDistance(Vector4) → void
```

```
create() → Boolean
```

➢ GeoJsonPickPointLayer

```
new Li.GeoJsonPickPointLayer()
```

属性：

```
radius : String

color : Color

heightOffset : String
```

实例方法：

```
create() → Boolean
```

➢ GeoJsonPolylineLayer

```
new Li.GeoJsonPolylineLayer()
```

属性：

```
color : Color
```

```
alpha : Number
```

```
width : Number
```

实例方法：

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
create() → Boolean
```

➢ GeoJsonPolygonLayer

```
new Li.GeoJsonPolygonLayer()
```

属性：

```
lineWidth : Number

lineColor : Color

fillColor : Color

lineAlpha : Number
```

```
fillAlpha : Number
```

```
pickColor :  Color
```

```
buildingAlpha : Number
```

```
entityType : Color
```

```
extrudeHeightField : String
```

```
floorsField : String
```

```
idField : String
```

```
nameField : String
```

实例方法：

```
setColorRamp(startColor, endColor, colorLevels) → void
```

参数名	类型	描述
startColor	Color	
endColor	Color	
colorLevels	Number	

```
setFloorHeight(height) → void
```

参数名	类型	描述
height	Number	

```
floorHeightGap() → Number
```

```
setFloorHeightGap(gap) → void
```

参数名	类型	描述
gap	Number	

```
altitude() → Number
```

```
setAltitude(height) → void
```

参数名	类型	描述
height	Number	

```
altitudeMethod() → AltitudeMethod
```

```
setAltitudeMethod(AltitudeMethod) → void
```

```
create() → Boolean
```

静态方法：

```
Li.GeoJsonPolygonLayer.getFloors() → Number
```

➢ GeoJsonPolygonQueryLayer

```
new Li.GeoJsonPolygonQueryLayer()
```

实例方法：

```
query(Vector3) → Vector2
```

```
query(Vector3) → Vector2
```

```
properties(id, key) → String
```

参数名	类型	描述
id	Number	
key	String	

```
create() → Boolean
```

➢ Heatmap

```
new Li.Heatmap()
```

属性：

```
radius : Number (readonly)

weightField : String (readonly)
```

实例方法：

```
create() → Boolean
```

➢ ExtrudeEntity

```
new Li.ExtrudeEntity()
```

属性：

```
extrudeHeight : Number
```

实例方法：

```
setOuter(Cartesian3Vector) → void
```

```
run() → void
```

```
createEntity(Material) → Entity
```

➢ ExtrudePipe

```
new Li.ExtrudePipe()
```

属性：

```
extrudeHeight : Number
```

实例方法：

```
setPositions(Cartesian3Vector) → void
```

```
setShape(Cartesian2Vector) → void
```

```
run() → void
```

```
createEntity(Material) → Entity
```

➢ GisUtil

静态方法：

```
Li.GisUtil.computeShape2D(radius, subdivision) → Cartesian2Vector
```

参数名	类型	描述
radius	Number	
subdivision	Number	

```
Li.GisUtil.computeShape2GeodeticSurface(radius, subdivision, center)
  → Cartesian3Vector
```

参数名	类型	描述
radius	Number	
subdivision	Number	
center	Vector3	

```
Li.GisUtil.calculateGeometryVolume(GeometryRenderer) →  void
```

Wasm 数据服务调用示例:

```
//var viewer = GlobalViewer;
//var scene = viewer.scene;
//var globe = scene.globe;
//var camera = scene.mainCamera;
//var cameraController = camera.cameraController;
```

//加载地形数据

```
Var addTerrain = function(globe, url)
{
    globe.setTerrainProviderUrl(url);
}
```

//设置默认地形

```
Var setDefaultTerrain = function(globe)
{
    globe.setDefaultTerrain();
}
```

//添加 arcgis 影像

```
Var addArcGisMapServerImageryLayer = function(globe)
{
    globe.addArcGisMapServerImageryLayer('https://services.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer');
}
```

//添加 wmts 底图示例

```
Var addWMTSLayer = function (globe)
{
        globe.lightingEnabled = false;
        var imageryLayer = globe.addWmsImageryLayer("IgnoreAxisOrientation=1&IgnoreGetMapUrl=1&IgnoreReportedLayerExtents=1&InvertAxisOrientation=1&crs=EPSG:4326&dpiMode=7&format=image/png&layers=wmts_4326_440300&styles=default&tileMatrixSet=c&url=https://jingzhe.szft.gov.cn/
```

```
sfmap/MapTileService/wmts?SERVICE%3DWMTS%26REQUEST%3DGetCapabilities%26STORETYPE%3D
merged-dat%26LAYER%3Dwmts_4326_440300%26PROJECTION%3D4326");
    return imageryLayer;
}
```

//添加影像

```
Var addImageryLayer = function(globe, url, rectangle, minLevel, maxLevel)
{
    if (minLevel < 0)
        minLevel = 0;
    if (maxLevel > 21)
        maxLevel = 21;

    // add WebMecrator imagery layer
    var imageryProvider = new Li.UrlTemplateImageryProvider(
      url,    // url
      true,   // useWebMercator
      maxLevel,     // maximumLevel
      minLevel,      // minimumLevel
      256,    // tileWidth
      256,    // tileHeight
      true    // hasAlphaChannel
    );
    globe.addImageryLayer(new Li.ImageryLayer(imageryProvider, rectangle));
}
```

// 添加天地图影像

```
Var addTiandituImagery = function ()
{
        var tianditu = new Li.TiandituImageryLayer();
        tianditu.url =
"https://t2.tianditu.gov.cn/DataServer?T=img_c&x={x}&y={y}&l={z}&tk=5ec3e996393521a
1a9fb1681f13201ff";
        tianditu.useWebMercator = false;
        tianditu.tileWidth = 256;
        tianditu.tileHeight = 256;
        tianditu.minimumLevel = 1;
        tianditu.maximumLevel = 17;
        tianditu.hasAlphaChannel = false;
        tianditu.isLabel = false;
        tianditu.componentComplete();
       return tianditu;
```

```
}
```

// 天地图区划

```
Var addTiandituZone = function ()
{
        var tianditu = new Li.TiandituImageryLayer();
        tianditu.url =
"https://t3.tianditu.gov.cn/DataServer?T=vec_c&x={x}&y={y}&l={z}&tk=5ec3e996393521a
1a9fb1681f13201ff";
        tianditu.useWebMercator = false;
        tianditu.tileWidth = 256;
        tianditu.tileHeight = 256;
        tianditu.minimumLevel = 1;
        tianditu.maximumLevel = 17;
        tianditu.hasAlphaChannel = false;
        tianditu.isLabel = false;
        tianditu.componentComplete();
        return tianditu;
}
```

//添加 tilesetLayer

```
Var addTilesetLayer = function (baseUrl)
{
        var tilesetLayer = new Li.TilesetLayer();
        //tilesetLayer.url = baseUrl + "data/0326/tileset.json";
        tilesetLayer.url = baseUrl + "data/tileset/tileset.json";
        //tilesetLayer.url = baseUrl + "data/tileset_skip2_1024/tileset.json";
        tilesetLayer.color = Li.Color.fromRgb(255, 251, 240, 255);
        // tilesetLayer.renderStyle = Li.RenderStyle.ColorStyle;
        tilesetLayer.clipLevelOfDetail = true;
        tilesetLayer.componentComplete();
        return tilesetLayer;
}
```

//加载 3Dtiles

```
Var load3DTiles = function (scene, url)
{
    var tileset = new Li.Tileset(url);
    // tileset.streamingMode = true;    // default 流式加载
    // tileset.skipLevelOfDetail = true; // default 跳过层级
```

```
    // tileset.genMeshNormals = false; // default   生成法线
    var entity = new Li.Entity();
    entity.addComponent(tileset);
    scene.addEntity(entity);
    return tileset;
}
```

//为 3Dtiles 添加遮罩

```
Var addFlattenMask = function(tileset, list)
{
    //小于等于 2，不能构成一个遮罩面
    if (list.length <= 2)
    {
        return;
    }

    var mask = new Li.FlattenMask();
    mask.maskHeight = 0; // maskHeight only useful for terrain mask

    mask.setPoints(list);
    tileset.addFlattenMask(mask);

    return mask;
}
```

//删除遮罩

```
Var removeFlatenMask = function (tileset, mask)
{
    tileset.removeFlattenMask(mask);
    mask.delete();
}
```

//加载 modellayer

```
Var modelLayer = function(url, carto, rotation, offset)
{
    let modelLayer = new Li.ModelLayer();
    modelLayer.url = url;
    modelLayer.cartographic = carto;
```

```
    modelLayer.rotation = rotation;
    modelLayer.offset = offset;
    modelLayer.componentComplete();

    return modelLayer;
}
```

// add gltf/obj/fbx/dae/... 模型

```
Var loadModel = function(scene, url, position, rotation)
{
    var model = new Li.Model(url);
    var entity = new Li.Entity();
    entity.transform.cartographic = position;   //坐标点
    entity.transform.rotation = rotation;       //旋转角度(四元数)
    entity.addComponent(model);
    scene.addEntity(entity);                    //添加到 rootEntity 的子节点中

    return entity;
}
```

//关闭大气效果

```
scene.atmosphere.inscatterAmount = 0;
```

//关闭地面受光

```
globe.lightingEnabled = false;


var addDZSB = function (baseUrl, tilesetLayer)
{
        var GeoJsonModel = new Li.GeoJsonModel();
        GeoJsonModel.fontSize = 10;
        GeoJsonModel.iconUrl = baseUrl + "symbols/images/哨兵.png";
        GeoJsonModel.selectedIconUrl = baseUrl + "symbols/images/哨兵 hover.png"; //
        GeoJsonModel.iconSize = Li.Vector2.create(48, 48);
        //GeoJsonModel.iconOffset = Li.Vector2.create(0, -12);
        GeoJsonModel.height = 3.0;
        //GeoJsonModel.heightField = "hight";
        //GeoJsonModel.addField("使用单位"); //需要获取的属性添加进去，可以添加多个属性
        GeoJsonModel.addField("FID");
```

```
        GeoJsonModel.addField("设备编号");
        GeoJsonModel.addField("街道");
        GeoJsonModel.addField("社区");

        //GeoJsonModel.labelField = "卡口";
        GeoJsonModel.labelField = "进出类型";   //文字标签字段
        GeoJsonModel.labelOffset = Li.Vector2.create(10, 12); //x, y 偏移值

        if (tilesetLayer != null && tilesetLayer != undefined)
            GeoJsonModel.setBase3DTileset(tilesetLayer.tileset());

        GeoJsonModel.load(baseUrl + "data/电子哨兵 0407.geojson"); //加载 geojson 文件
        //or
        //GeoJsonModel.addString(Utils.testString);   //添加 string (geojson 格式
string)
       return GeoJsonModel;
}
```

//鼠标点击 拾取物体 / 拾取标签属性

```
document.onclick = function (event)
{
    var e = event || window.event;
    var posX = 0, posY = 0;
    posX = e.clientX;
    posY = e.clientY;

    if (e)
    {
         let feature = Li.GeoJsonModel.getSelectedFeature();
        if (feature)
        {
            console.log("pick feature");
            //let property = feature.getProperty("使用单位"); //要查询的属性
            let property = feature.getProperty("设备编号"); //要查询的属性
            if (property)
            {
                console.log("设备编号: " + property);
            }
        }

    }
}
```