Class List

Classes Methods Properties Files
Search:

- Top Level Namespace

Index (D) » AWS » DynamoDB » DocumentClient

Class: AWS.DynamoDB.DocumentClient

Inherits:
Object

- Object
- AWS.DynamoDB.DocumentClient

show all

Defined in:
lib/dynamodb/document_client.js

Overview

The document client simplifies working with items in Amazon DynamoDB by abstracting away the notion of attribute values. This abstraction annotates native JavaScript types supplied as input parameters, as well as converts annotated response data to native JavaScript types.

Marshalling Input and Unmarshalling Response Data

The document client affords developers the use of native JavaScript types instead of `AttributeValues` to simplify the JavaScript development experience with Amazon DynamoDB. JavaScript objects passed in as parameters are marshalled into `AttributeValue` shapes required by Amazon DynamoDB. Responses from DynamoDB are unmarshalled into plain JavaScript objects by the `DocumentClient`. The `DocumentClient`, does not accept `AttributeValues` in favor of native JavaScript types.

JavaScript Type	DynamoDB AttributeValue
String	S
Number	N
Boolean	BOOL
null	NULL
Array	L
Object	M
Buffer, File, Blob, ArrayBuffer, DataView, and JavaScript typed arrays	B

Support for Sets

The `DocumentClient` offers a convenient way to create sets from JavaScript Arrays. The type of set is inferred from the first element in the array. DynamoDB supports string, number, and binary sets. To learn more about supported types see the Amazon DynamoDB Data Model Documentation For more information see createSet()

Constructor Summary collapse

- new **AWS.DynamoDB.DocumentClient**(options) ⇒ void constructor

  Creates a DynamoDB document client with a set of configuration options.

Method Summary collapse

- **batchGet**(params, callback) ⇒ AWS.Request

  Returns the attributes of one or more items from one or more tables by delegating to `AWS.DynamoDB.batchGetItem()`.

- **batchWrite**(params, callback) ⇒ AWS.Request

  Puts or deletes multiple items in one or more tables by delegating to `AWS.DynamoDB.batchWriteItem()`.

- **createSet**(list, options) ⇒ void

  Creates a set of elements inferring the type of set from the type of the first element.

- **delete**(params, callback) ⇒ AWS.Request

  Deletes a single item in a table by primary key by delegating to `AWS.DynamoDB.deleteItem()`.

- **get**(params, callback) ⇒ AWS.Request

  Returns a set of attributes for the item with the given primary key by delegating to `AWS.DynamoDB.getItem()`.

- **put**(params, callback) ⇒ AWS.Request

  Creates a new item, or replaces an old item with a new item by delegating to `AWS.DynamoDB.putItem()`.

- **query**(params, callback) ⇒ AWS.Request

  Directly access items from a table by primary key or a secondary index.

- **scan**(params, callback) ⇒ AWS.Request

  Returns one or more items and item attributes by accessing every item in a table or a secondary index.

- **update**(params, callback) ⇒ AWS.Request

  Edits an existing item's attributes, or adds a new item to the table if it does not already exist by delegating to `AWS.DynamoDB.updateItem()`.

Constructor Details

new AWS.DynamoDB.DocumentClient(options) ⇒ `void`

Creates a DynamoDB document client with a set of configuration options.

Options Hash (`options`):

- params (`map`) —

  An optional map of parameters to bind to every request sent by this service object.

- service (`AWS.DynamoDB`) —

  An optional pre-configured instance of the AWS.DynamoDB service object to use for requests. The object may bound parameters used by the document client.

See Also:

- AWS.DynamoDB.constructor()

Method Details

batchGet(params, callback) ⇒ `AWS.Request`

Returns the attributes of one or more items from one or more tables by delegating to `AWS.DynamoDB.batchGetItem()`.

Supply the same parameters as AWS.DynamoDB.batchGetItem() with `AttributeValues` substituted by native JavaScript types.

Examples:

Get items from multiple tables

```
var params = {
  RequestItems: {
    'Table-1': {
      Keys: [
        {
           HashKey: 'haskey',
           NumberRangeKey: 1
        }
      ]
    },
    'Table-2': {
      Keys: [
        { foo: 'bar' },
      ]
    }
  }
};

var docClient = new AWS.DynamoDB.DocumentClient();

docClient.batchGet(params, function(err, data) {
  if (err) console.log(err);
  else console.log(data);
});
```

Calling the batchGet operation

```
var params = {
  RequestItems: { /* required */
    someKey: {
      Keys: [ /* required */
        {
          someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
          /* anotherKey: ... */
        },
        /* more items */
      ],
      AttributesToGet: [
        'STRING_VALUE',
        /* more items */
      ],
      ConsistentRead: true || false,
      ExpressionAttributeNames: {
        someKey: 'STRING_VALUE',
        /* anotherKey: ... */
      },
      ProjectionExpression: 'STRING_VALUE'
    },
    /* anotherKey: ... */
  },
  ReturnConsumedCapacity: 'INDEXES | TOTAL | NONE'
};
documentclient.batchGet(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Parameters:

- params (`Object`) —
  - `RequestItems` — (`map<map>`)
    - `Keys` — **required** — (`Array<map<map>>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `AttributesToGet` — (`Array<String>`)
    - `ConsistentRead` — (`Boolean`)
    - `ProjectionExpression` — (`String`)
    - `ExpressionAttributeNames` — (`map<String>`)
  - `ReturnConsumedCapacity` — (`String`) Possible values include:
    - `"INDEXES"`
    - `"TOTAL"`
    - `"NONE"`

Callback (`callback`):

- **function**(err, data) { ... }

  Called when a response from the service is returned. If a callback is not supplied, you must call AWS.Request.send() on the returned request object to initiate the request.

  Context (`this`):

  - (`AWS.Response`) —

    the response object containing error, data properties, and the original request object.

  Parameters:

  - err (`Error`) —

    the error object returned from the request. Set to `null` if the request is successful.

  - data (`Object`) —

the de-serialized data returned from the request. Set to `null` if a request error occurs. The `data` object has the following properties:

- `Responses` — (`map<Array<map<map>>>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
- `UnprocessedKeys` — (`map<map>`)
  - `Keys` — **required** — (`Array<map<map>>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `AttributesToGet` — (`Array<String>`)
  - `ConsistentRead` — (`Boolean`)
  - `ProjectionExpression` — (`String`)
  - `ExpressionAttributeNames` — (`map<String>`)
- `ConsumedCapacity` — (`Array<map>`)
  - `TableName` — (`String`)
  - `CapacityUnits` — (`Float`)
  - `Table` — (`map`)
    - `CapacityUnits` — (`Float`)
  - `LocalSecondaryIndexes` — (`map<map>`)
    - `CapacityUnits` — (`Float`)
  - `GlobalSecondaryIndexes` — (`map<map>`)
    - `CapacityUnits` — (`Float`)

Returns:

- (`AWS.Request`) —

  a handle to the operation request for subsequent event callback registration.

See Also:

- AWS.DynamoDB.batchGetItem()

batchWrite(params, callback) ⇒ `AWS.Request`

Puts or deletes multiple items in one or more tables by delegating to `AWS.DynamoDB.batchWriteItem()`.

Supply the same parameters as AWS.DynamoDB.batchWriteItem() with `AttributeValues` substituted by native JavaScript types.

Examples:

Write to and delete from a table

```
var params = {
  RequestItems: {
    'Table-1': [
      {
        DeleteRequest: {
          Key: { HashKey: 'someKey' }
        }
      },
      {
        PutRequest: {
          Item: {
            HashKey: 'anotherKey',
            NumAttribute: 1,
            BoolAttribute: true,
            ListAttribute: [1, 'two', false],
            MapAttribute: { foo: 'bar' }
          }
        }
      }
    ]
  }
};

var docClient = new AWS.DynamoDB.DocumentClient();

docClient.batchWrite(params, function(err, data) {
  if (err) console.log(err);
  else console.log(data);
});
```

Calling the batchWrite operation

```
var params = {
  RequestItems: { /* required */
    someKey: [
      {
        DeleteRequest: {
          Key: { /* required */
            someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
            /* anotherKey: ... */
          }
        },
        PutRequest: {
          Item: { /* required */
            someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
            /* anotherKey: ... */
          }
        }
      },
      /* more items */
    ],
    /* anotherKey: ... */
  },
  ReturnConsumedCapacity: 'INDEXES | TOTAL | NONE',
  ReturnItemCollectionMetrics: 'SIZE | NONE'
};
documentclient.batchWrite(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Parameters:

- params (`Object`) —
  - `RequestItems` — (`map<Array<map>>`)
    - `PutRequest` — (`map`)
      - `Item` — **required** — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `DeleteRequest` — (`map`)
      - `Key` — **required** — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `ReturnConsumedCapacity` — (`String`) Possible values include:
    - `"INDEXES"`
    - `"TOTAL"`
    - `"NONE"`
  - `ReturnItemCollectionMetrics` — (`String`) Possible values include:
    - `"SIZE"`
    - `"NONE"`

Callback (`callback`):

- **function**(err, data) { ... }

  Called when a response from the service is returned. If a callback is not supplied, you must call AWS.Request.send() on the returned request object to initiate the request.

  Context (`this`):

  - (`AWS.Response`) —

    the response object containing error, data properties, and the original request object.

  Parameters:

  - err (`Error`) —

    the error object returned from the request. Set to `null` if the request is successful.

  - data (`Object`) —

    the de-serialized data returned from the request. Set to `null` if a request error occurs. The `data` object has the following properties:

    - `UnprocessedItems` — (`map<Array<map>>`)
      - `PutRequest` — (`map`)
        - `Item` — **required** — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
      - `DeleteRequest` — (`map`)
        - `Key` — **required** — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ItemCollectionMetrics` — (`map<Array<map>>`)
      - `ItemCollectionKey` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
      - `SizeEstimateRangeGB` — (`Array<Float>`)
    - `ConsumedCapacity` — (`Array<map>`)
      - `TableName` — (`String`)
      - `CapacityUnits` — (`Float`)
      - `Table` — (`map`)
        - `CapacityUnits` — (`Float`)
      - `LocalSecondaryIndexes` — (`map<map>`)
        - `CapacityUnits` — (`Float`)
      - `GlobalSecondaryIndexes` — (`map<map>`)
        - `CapacityUnits` — (`Float`)

Returns:

- (`AWS.Request`) —

  a handle to the operation request for subsequent event callback registration.

See Also:

- AWS.DynamoDB.batchWriteItem()

createSet(list, options) ⇒ `void`

Creates a set of elements inferring the type of set from the type of the first element. Amazon DynamoDB currently supports the number sets, string sets, and binary sets. For more information about DynamoDB data types see the documentation on the Amazon DynamoDB Data Model.

Examples:

Creating a number set

```
var docClient = new AWS.DynamoDB.DocumentClient();

var params = {
  Item: {
    hashkey: 'hashkey'
    numbers: docClient.createSet([1, 2, 3]);
  }
};

docClient.put(params, function(err, data) {
  if (err) console.log(err);
  else console.log(data);
});
```

Parameters:

- list (`Array`) —

  Collection to represent your DynamoDB Set

- options (`map`) —
  - **validate** [Boolean] set to true if you want to validate the type of each element in the set. Defaults to `false`.

delete(params, callback) ⇒ `AWS.Request`

Deletes a single item in a table by primary key by delegating to `AWS.DynamoDB.deleteItem()`

Supply the same parameters as AWS.DynamoDB.deleteItem() with `AttributeValues` substituted by native JavaScript types.

Examples:

Delete an item from a table

```
var params = {
  TableName : 'Table',
  Key: {
    HashKey: 'hashkey',
    NumberRangeKey: 1
  }
};

var docClient = new AWS.DynamoDB.DocumentClient();

docClient.delete(params, function(err, data) {
  if (err) console.log(err);
  else console.log(data);
});
```

Calling the delete operation

```
var params = {
  Key: { /* required */
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  TableName: 'STRING_VALUE', /* required */
  ConditionExpression: 'STRING_VALUE',
  ConditionalOperator: 'AND | OR',
  Expected: {
    someKey: {
      AttributeValueList: [
        someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
        /* more items */
      ],
      ComparisonOperator: 'EQ | NE | IN | LE | LT | GE | GT | BETWEEN | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS | BEGINS_WITH',
      Exists: true || false,
      Value: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */
    },
    /* anotherKey: ... */
  },
  ExpressionAttributeNames: {
    someKey: 'STRING_VALUE',
    /* anotherKey: ... */
  },
  ExpressionAttributeValues: {
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  ReturnConsumedCapacity: 'INDEXES | TOTAL | NONE',
  ReturnItemCollectionMetrics: 'SIZE | NONE',
  ReturnValues: 'NONE | ALL_OLD | UPDATED_OLD | ALL_NEW | UPDATED_NEW'
};
documentclient.delete(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Parameters:

- params (`Object`) —
  - `TableName` — (`String`)
  - `Key` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `Expected` — (`map<map>`)
    - `Value` — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `Exists` — (`Boolean`)
    - `ComparisonOperator` — (`String`) Possible values include:
      - `"EQ"`
      - `"NE"`
      - `"IN"`
      - `"LE"`
      - `"LT"`
      - `"GE"`
      - `"GT"`
      - `"BETWEEN"`
      - `"NOT_NULL"`
      - `"NULL"`
      - `"CONTAINS"`
      - `"NOT_CONTAINS"`
      - `"BEGINS_WITH"`
    - `AttributeValueList` — (`Array<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `ConditionalOperator` — (`String`) Possible values include:
    - `"AND"`
    - `"OR"`
  - `ReturnValues` — (`String`) Possible values include:
    - `"NONE"`
    - `"ALL_OLD"`
    - `"UPDATED_OLD"`
    - `"ALL_NEW"`
    - `"UPDATED_NEW"`
  - `ReturnConsumedCapacity` — (`String`) Possible values include:
    - `"INDEXES"`
    - `"TOTAL"`
    - `"NONE"`
  - `ReturnItemCollectionMetrics` — (`String`) Possible values include:
    - `"SIZE"`
    - `"NONE"`
  - `ConditionExpression` — (`String`)
  - `ExpressionAttributeNames` — (`map<String>`)
  - `ExpressionAttributeValues` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model

Callback (`callback`):

- **function**(err, data) { ... }

  Called when a response from the service is returned. If a callback is not supplied, you must call AWS.Request.send() on the returned request object to initiate the request.

  Context (`this`):

  - (`AWS.Response`) —

    the response object containing error, data properties, and the original request object.

  Parameters:

- err (`Error`) —

  the error object returned from the request. Set to `null` if the request is successful.

- data (`Object`) —

  the de-serialized data returned from the request. Set to `null` if a request error occurs. The `data` object has the following properties:

  - `Attributes` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `ConsumedCapacity` — (`map`)
    - `TableName` — (`String`)
    - `CapacityUnits` — (`Float`)
    - `Table` — (`map`)
      - `CapacityUnits` — (`Float`)
    - `LocalSecondaryIndexes` — (`map<map>`)
      - `CapacityUnits` — (`Float`)
    - `GlobalSecondaryIndexes` — (`map<map>`)
      - `CapacityUnits` — (`Float`)
  - `ItemCollectionMetrics` — (`map`)
    - `ItemCollectionKey` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `SizeEstimateRangeGB` — (`Array<Float>`)

Returns:

- (`AWS.Request`) —

  a handle to the operation request for subsequent event callback registration.

See Also:

- AWS.DynamoDB.deleteItem()

get(params, callback) ⇒ AWS.Request

Returns a set of attributes for the item with the given primary key by delegating to `AWS.DynamoDB.getItem()`.

Supply the same parameters as AWS.DynamoDB.getItem() with `AttributeValues` substituted by native JavaScript types.

Examples:

Get an item from a table

```
var params = {
  TableName : 'Table',
  Key: {
    HashKey: 'hashkey'
  }
};

var docClient = new AWS.DynamoDB.DocumentClient();

docClient.get(params, function(err, data) {
  if (err) console.log(err);
  else console.log(data);
});
```

Calling the get operation

```
var params = {
  Key: { /* required */
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  TableName: 'STRING_VALUE', /* required */
  AttributesToGet: [
    'STRING_VALUE',
    /* more items */
  ],
  ConsistentRead: true || false,
  ExpressionAttributeNames: {
    someKey: 'STRING_VALUE',
    /* anotherKey: ... */
  },
  ProjectionExpression: 'STRING_VALUE',
  ReturnConsumedCapacity: 'INDEXES | TOTAL | NONE'
};
documentclient.get(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Parameters:

- params (`Object`) —
  - `TableName` — (`String`)
  - `Key` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `AttributesToGet` — (`Array<String>`)
  - `ConsistentRead` — (`Boolean`)
  - `ReturnConsumedCapacity` — (`String`) Possible values include:
    - `"INDEXES"`
    - `"TOTAL"`
    - `"NONE"`
  - `ProjectionExpression` — (`String`)
  - `ExpressionAttributeNames` — (`map<String>`)

Callback (`callback`):

- **function**(err, data) { ... }

  Called when a response from the service is returned. If a callback is not supplied, you must call AWS.Request.send() on the returned request object to initiate the request.

  Context (`this`):

  - (`AWS.Response`) —

    the response object containing error, data properties, and the original request object.

  Parameters:

  - err (`Error`) —

    the error object returned from the request. Set to `null` if the request is successful.

  - data (`Object`) —

    the de-serialized data returned from the request. Set to `null` if a request error occurs. The `data` object has the following properties:

    - `Item` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ConsumedCapacity` — (`map`)
      - `TableName` — (`String`)
      - `CapacityUnits` — (`Float`)
      - `Table` — (`map`)
        - `CapacityUnits` — (`Float`)
      - `LocalSecondaryIndexes` — (`map<map>`)
        - `CapacityUnits` — (`Float`)
      - `GlobalSecondaryIndexes` — (`map<map>`)
        - `CapacityUnits` — (`Float`)

Returns:

- (`AWS.Request`) —

  a handle to the operation request for subsequent event callback registration.

See Also:

- AWS.DynamoDB.getItem()

put(params, callback) ⇒ `AWS.Request`

Creates a new item, or replaces an old item with a new item by delegating to `AWS.DynamoDB.putItem()`.

Supply the same parameters as AWS.DynamoDB.putItem() with `AttributeValues` substituted by native JavaScript types.

Examples:

Create a new item in a table

```
var params = {
  TableName : 'Table',
  Item: {
     HashKey: 'haskey',
     NumAttribute: 1,
     BoolAttribute: true,
     ListAttribute: [1, 'two', false],
     MapAttribute: { foo: 'bar'},
     NullAttribute: null
  }
};
```

```
var docClient = new AWS.DynamoDB.DocumentClient();

docClient.put(params, function(err, data) {
  if (err) console.log(err);
  else console.log(data);
});
```

Calling the put operation

```
var params = {
  Item: { /* required */
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  TableName: 'STRING_VALUE', /* required */
  ConditionExpression: 'STRING_VALUE',
  ConditionalOperator: 'AND | OR',
  Expected: {
    someKey: {
      AttributeValueList: [
        someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
        /* more items */
      ],
      ComparisonOperator: 'EQ | NE | IN | LE | LT | GE | GT | BETWEEN | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS | BEGINS_WITH',
      Exists: true || false,
      Value: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */
    },
    /* anotherKey: ... */
  },
  ExpressionAttributeNames: {
    someKey: 'STRING_VALUE',
    /* anotherKey: ... */
  },
  ExpressionAttributeValues: {
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  ReturnConsumedCapacity: 'INDEXES | TOTAL | NONE',
  ReturnItemCollectionMetrics: 'SIZE | NONE',
  ReturnValues: 'NONE | ALL_OLD | UPDATED_OLD | ALL_NEW | UPDATED_NEW'
};
documentclient.put(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Parameters:

- params (`Object`) —
  - `TableName` — (`String`)
  - `Item` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `Expected` — (`map<map>`)
    - `Value` — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `Exists` — (`Boolean`)
    - `ComparisonOperator` — (`String`) Possible values include:
      - `"EQ"`
      - `"NE"`
      - `"IN"`
      - `"LE"`
      - `"LT"`
      - `"GE"`
      - `"GT"`
      - `"BETWEEN"`
      - `"NOT_NULL"`
      - `"NULL"`
      - `"CONTAINS"`
      - `"NOT_CONTAINS"`
      - `"BEGINS_WITH"`
    - `AttributeValueList` — (`Array<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `ReturnValues` — (`String`) Possible values include:
    - `"NONE"`
    - `"ALL_OLD"`
    - `"UPDATED_OLD"`
    - `"ALL_NEW"`
    - `"UPDATED_NEW"`
  - `ReturnConsumedCapacity` — (`String`) Possible values include:
    - `"INDEXES"`
    - `"TOTAL"`
    - `"NONE"`
  - `ReturnItemCollectionMetrics` — (`String`) Possible values include:
    - `"SIZE"`
    - `"NONE"`

- `ConditionalOperator` — (`String`) Possible values include:
  - `"AND"`
  - `"OR"`
- `ConditionExpression` — (`String`)
- `ExpressionAttributeNames` — (`map<String>`)
- `ExpressionAttributeValues` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model

Callback (`callback`):

- **function**(err, data) { ... }

  Called when a response from the service is returned. If a callback is not supplied, you must call AWS.Request.send() on the returned request object to initiate the request.

  Context (`this`):

  - (`AWS.Response`) —

    the response object containing error, data properties, and the original request object.

  Parameters:

  - err (`Error`) —

    the error object returned from the request. Set to `null` if the request is successful.

  - data (`Object`) —

    the de-serialized data returned from the request. Set to `null` if a request error occurs. The `data` object has the following properties:

    - `Attributes` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ConsumedCapacity` — (`map`)
      - `TableName` — (`String`)
      - `CapacityUnits` — (`Float`)
      - `Table` — (`map`)
        - `CapacityUnits` — (`Float`)
      - `LocalSecondaryIndexes` — (`map<map>`)
        - `CapacityUnits` — (`Float`)
      - `GlobalSecondaryIndexes` — (`map<map>`)
        - `CapacityUnits` — (`Float`)
    - `ItemCollectionMetrics` — (`map`)
      - `ItemCollectionKey` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
      - `SizeEstimateRangeGB` — (`Array<Float>`)

Returns:

- (`AWS.Request`) —

  a handle to the operation request for subsequent event callback registration.

See Also:

- AWS.DynamoDB.putItem()

query(params, callback) ⇒ `AWS.Request`

Directly access items from a table by primary key or a secondary index.

Supply the same parameters as AWS.DynamoDB.query() with `AttributeValues` substituted by native JavaScript types.

Examples:

Query an index

```
var params = {
  TableName: 'Table',
  IndexName: 'Index',
  KeyConditionExpression: 'HashKey = :hkey and RangeKey > :rkey',
  ExpressionAttributeValues: {
    ':hkey': 'key',
    ':rkey': 2015
  }
};

var docClient = new AWS.DynamoDB.DocumentClient();

docClient.query(params, function(err, data) {
   if (err) console.log(err);
   else console.log(data);
});
```

Calling the query operation

```
var params = {
  TableName: 'STRING_VALUE', /* required */
  AttributesToGet: [
    'STRING_VALUE',
    /* more items */
  ],
  ConditionalOperator: 'AND | OR',
  ConsistentRead: true || false,
  ExclusiveStartKey: {
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  ExpressionAttributeNames: {
    someKey: 'STRING_VALUE',
    /* anotherKey: ... */
  },
  ExpressionAttributeValues: {
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  FilterExpression: 'STRING_VALUE',
  IndexName: 'STRING_VALUE',
  KeyConditionExpression: 'STRING_VALUE',
  KeyConditions: {
    someKey: {
      ComparisonOperator: 'EQ | NE | IN | LE | LT | GE | GT | BETWEEN | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS | BEGINS_WITH', /* required */
      AttributeValueList: [
        someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
        /* more items */
      ]
    },
    /* anotherKey: ... */
  },
  Limit: 0,
  ProjectionExpression: 'STRING_VALUE',
  QueryFilter: {
    someKey: {
      ComparisonOperator: 'EQ | NE | IN | LE | LT | GE | GT | BETWEEN | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS | BEGINS_WITH', /* required */
      AttributeValueList: [
        someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
        /* more items */
      ]
    },
    /* anotherKey: ... */
  },
  ReturnConsumedCapacity: 'INDEXES | TOTAL | NONE',
  ScanIndexForward: true || false,
  Select: 'ALL_ATTRIBUTES | ALL_PROJECTED_ATTRIBUTES | SPECIFIC_ATTRIBUTES | COUNT'
};
documentclient.query(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Parameters:

- params (`Object`) —
  - `TableName` — (`String`)
  - `IndexName` — (`String`)
  - `Select` — (`String`) Possible values include:
    - `"ALL_ATTRIBUTES"`
    - `"ALL_PROJECTED_ATTRIBUTES"`
    - `"SPECIFIC_ATTRIBUTES"`
    - `"COUNT"`
  - `AttributesToGet` — (`Array<String>`)
  - `Limit` — (`Integer`)
  - `ConsistentRead` — (`Boolean`)
  - `KeyConditions` — (`map<map>`)
    - `AttributeValueList` — (`Array<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ComparisonOperator` — **required** — (`String`) Possible values include:
      - `"EQ"`
      - `"NE"`
      - `"IN"`
      - `"LE"`
      - `"LT"`
      - `"GE"`
      - `"GT"`
      - `"BETWEEN"`
      - `"NOT_NULL"`
      - `"NULL"`

        - `"CONTAINS"`
        - `"NOT_CONTAINS"`
        - `"BEGINS_WITH"`
  - `QueryFilter` — (`map<map>`)
    - `AttributeValueList` — (`Array<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ComparisonOperator` — **required** — (`String`) Possible values include:
      - `"EQ"`
      - `"NE"`
      - `"IN"`
      - `"LE"`
      - `"LT"`
      - `"GE"`
      - `"GT"`
      - `"BETWEEN"`
      - `"NOT_NULL"`
      - `"NULL"`
      - `"CONTAINS"`
      - `"NOT_CONTAINS"`
      - `"BEGINS_WITH"`
  - `ConditionalOperator` — (`String`) Possible values include:
    - `"AND"`
    - `"OR"`
  - `ScanIndexForward` — (`Boolean`)
  - `ExclusiveStartKey` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `ReturnConsumedCapacity` — (`String`) Possible values include:
    - `"INDEXES"`
    - `"TOTAL"`
    - `"NONE"`
  - `ProjectionExpression` — (`String`)
  - `FilterExpression` — (`String`)
  - `KeyConditionExpression` — (`String`)
  - `ExpressionAttributeNames` — (`map<String>`)
  - `ExpressionAttributeValues` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model

Callback (`callback`):

- **function**(err, data) { ... }

  Called when a response from the service is returned. If a callback is not supplied, you must call AWS.Request.send() on the returned request object to initiate the request.

  Context (`this`):

  - (`AWS.Response`) —

    the response object containing error, data properties, and the original request object.

  Parameters:

  - err (`Error`) —

    the error object returned from the request. Set to `null` if the request is successful.

  - data (`Object`) —

    the de-serialized data returned from the request. Set to `null` if a request error occurs. The `data` object has the following properties:

    - `Items` — (`Array<map<map>>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `Count` — (`Integer`)
    - `ScannedCount` — (`Integer`)
    - `LastEvaluatedKey` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ConsumedCapacity` — (`map`)
      - `TableName` — (`String`)
      - `CapacityUnits` — (`Float`)
      - `Table` — (`map`)
        - `CapacityUnits` — (`Float`)
      - `LocalSecondaryIndexes` — (`map<map>`)
        - `CapacityUnits` — (`Float`)
      - `GlobalSecondaryIndexes` — (`map<map>`)
        - `CapacityUnits` — (`Float`)

Returns:

- (`AWS.Request`) —

  a handle to the operation request for subsequent event callback registration.

See Also:

- AWS.DynamoDB.query()

scan(params, callback) ⇒ AWS.Request

Returns one or more items and item attributes by accessing every item in a table or a secondary index.

Supply the same parameters as AWS.DynamoDB.scan() with `AttributeValues` substituted by native JavaScript types.

Examples:

Scan the table with a filter expression

```
var params = {
  TableName : 'Table',
  FilterExpression : 'Year = :this_year',
  ExpressionAttributeValues : {':this_year' : 2015}
};

var docClient = new AWS.DynamoDB.DocumentClient();

docClient.scan(params, function(err, data) {
   if (err) console.log(err);
   else console.log(data);
});
```

Calling the scan operation

```
var params = {
  TableName: 'STRING_VALUE', /* required */
  AttributesToGet: [
    'STRING_VALUE',
    /* more items */
  ],
  ConditionalOperator: 'AND | OR',
  ConsistentRead: true || false,
  ExclusiveStartKey: {
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  ExpressionAttributeNames: {
    someKey: 'STRING_VALUE',
    /* anotherKey: ... */
  },
  ExpressionAttributeValues: {
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  FilterExpression: 'STRING_VALUE',
  IndexName: 'STRING_VALUE',
  Limit: 0,
  ProjectionExpression: 'STRING_VALUE',
  ReturnConsumedCapacity: 'INDEXES | TOTAL | NONE',
  ScanFilter: {
    someKey: {
      ComparisonOperator: 'EQ | NE | IN | LE | LT | GE | GT | BETWEEN | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS | BEGINS_WITH', /* required */
      AttributeValueList: [
        someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
        /* more items */
      ]
    },
    /* anotherKey: ... */
  },
  Segment: 0,
  Select: 'ALL_ATTRIBUTES | ALL_PROJECTED_ATTRIBUTES | SPECIFIC_ATTRIBUTES | COUNT',
  TotalSegments: 0
};
documentclient.scan(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Parameters:

- params (`Object`) —
  - `TableName` — (`String`)
  - `IndexName` — (`String`)
  - `AttributesToGet` — (`Array<String>`)
  - `Limit` — (`Integer`)
  - `Select` — (`String`) Possible values include:
    - `"ALL_ATTRIBUTES"`
    - `"ALL_PROJECTED_ATTRIBUTES"`
    - `"SPECIFIC_ATTRIBUTES"`

        - `"COUNT"`
    - `ScanFilter` — (`map<map>`)
        - `AttributeValueList` — (`Array<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
        - `ComparisonOperator` — **required** — (`String`) Possible values include:
            - `"EQ"`
            - `"NE"`
            - `"IN"`
            - `"LE"`
            - `"LT"`
            - `"GE"`
            - `"GT"`
            - `"BETWEEN"`
            - `"NOT_NULL"`
            - `"NULL"`
            - `"CONTAINS"`
            - `"NOT_CONTAINS"`
            - `"BEGINS_WITH"`
    - `ConditionalOperator` — (`String`) Possible values include:
        - `"AND"`
        - `"OR"`
    - `ExclusiveStartKey` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ReturnConsumedCapacity` — (`String`) Possible values include:
        - `"INDEXES"`
        - `"TOTAL"`
        - `"NONE"`
    - `TotalSegments` — (`Integer`)
    - `Segment` — (`Integer`)
    - `ProjectionExpression` — (`String`)
    - `FilterExpression` — (`String`)
    - `ExpressionAttributeNames` — (`map<String>`)
    - `ExpressionAttributeValues` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ConsistentRead` — (`Boolean`)

Callback (`callback`):

- **function**(err, data) { ... }

  Called when a response from the service is returned. If a callback is not supplied, you must call AWS.Request.send() on the returned request object to initiate the request.

  Context (`this`):

    - (`AWS.Response`) —

      the response object containing error, data properties, and the original request object.

  Parameters:

    - err (`Error`) —

      the error object returned from the request. Set to `null` if the request is successful.

    - data (`Object`) —

      the de-serialized data returned from the request. Set to `null` if a request error occurs. The `data` object has the following properties:

        - `Items` — (`Array<map<map>>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
        - `Count` — (`Integer`)
        - `ScannedCount` — (`Integer`)
        - `LastEvaluatedKey` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
        - `ConsumedCapacity` — (`map`)
            - `TableName` — (`String`)
            - `CapacityUnits` — (`Float`)
            - `Table` — (`map`)
                - `CapacityUnits` — (`Float`)
            - `LocalSecondaryIndexes` — (`map<map>`)
                - `CapacityUnits` — (`Float`)
            - `GlobalSecondaryIndexes` — (`map<map>`)
                - `CapacityUnits` — (`Float`)

Returns:

- (`AWS.Request`) —

  a handle to the operation request for subsequent event callback registration.

See Also:

- AWS.DynamoDB.scan()

update(params, callback) ⇒ AWS.Request

Edits an existing item's attributes, or adds a new item to the table if it does not already exist by delegating to `AWS.DynamoDB.updateItem()`.

Supply the same parameters as AWS.DynamoDB.updateItem() with `AttributeValues` substituted by native JavaScript types.

Examples:

Update an item with expressions

```
var params = {
  TableName: 'Table',
  Key: { HashKey : 'hashkey' },
  UpdateExpression: 'set #a = :x + :y',
  ConditionExpression: '#a < :MAX',
  ExpressionAttributeNames: {'#a' : 'Sum'},
  ExpressionAttributeValues: {
    ':x' : 20,
    ':y' : 45,
    ':MAX' : 100,
  }
};

var docClient = new AWS.DynamoDB.DocumentClient();

docClient.update(params, function(err, data) {
   if (err) console.log(err);
   else console.log(data);
});
```

Calling the update operation

```
var params = {
  Key: { /* required */
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  TableName: 'STRING_VALUE', /* required */
  AttributeUpdates: {
    someKey: {
      Action: 'ADD | PUT | DELETE',
      Value: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */
    },
    /* anotherKey: ... */
  },
  ConditionExpression: 'STRING_VALUE',
  ConditionalOperator: 'AND | OR',
  Expected: {
    someKey: {
      AttributeValueList: [
        someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
        /* more items */
      ],
      ComparisonOperator: 'EQ | NE | IN | LE | LT | GE | GT | BETWEEN | NOT_NULL | NULL | CONTAINS | NOT_CONTAINS | BEGINS_WITH',
      Exists: true || false,
      Value: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */
    },
    /* anotherKey: ... */
  },
  ExpressionAttributeNames: {
    someKey: 'STRING_VALUE',
    /* anotherKey: ... */
  },
  ExpressionAttributeValues: {
    someKey: someValue /* "str" | 10 | true | false | null | [1, "a"] | {a: "b"} */,
    /* anotherKey: ... */
  },
  ReturnConsumedCapacity: 'INDEXES | TOTAL | NONE',
  ReturnItemCollectionMetrics: 'SIZE | NONE',
  ReturnValues: 'NONE | ALL_OLD | UPDATED_OLD | ALL_NEW | UPDATED_NEW',
  UpdateExpression: 'STRING_VALUE'
};
documentclient.update(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Parameters:

- params (`Object`) —
  - `TableName` — (`String`)

- `Key` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
- `AttributeUpdates` — (`map<map>`)
  - `Value` — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `Action` — (`String`) Possible values include:
    - `"ADD"`
    - `"PUT"`
    - `"DELETE"`
- `Expected` — (`map<map>`)
  - `Value` — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
  - `Exists` — (`Boolean`)
  - `ComparisonOperator` — (`String`) Possible values include:
    - `"EQ"`
    - `"NE"`
    - `"IN"`
    - `"LE"`
    - `"LT"`
    - `"GE"`
    - `"GT"`
    - `"BETWEEN"`
    - `"NOT_NULL"`
    - `"NULL"`
    - `"CONTAINS"`
    - `"NOT_CONTAINS"`
    - `"BEGINS_WITH"`
  - `AttributeValueList` — (`Array<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
- `ConditionalOperator` — (`String`) Possible values include:
  - `"AND"`
  - `"OR"`
- `ReturnValues` — (`String`) Possible values include:
  - `"NONE"`
  - `"ALL_OLD"`
  - `"UPDATED_OLD"`
  - `"ALL_NEW"`
  - `"UPDATED_NEW"`
- `ReturnConsumedCapacity` — (`String`) Possible values include:
  - `"INDEXES"`
  - `"TOTAL"`
  - `"NONE"`
- `ReturnItemCollectionMetrics` — (`String`) Possible values include:
  - `"SIZE"`
  - `"NONE"`
- `UpdateExpression` — (`String`)
- `ConditionExpression` — (`String`)
- `ExpressionAttributeNames` — (`map<String>`)
- `ExpressionAttributeValues` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model

Callback (`callback`):

- **function**(err, data) { ... }

  Called when a response from the service is returned. If a callback is not supplied, you must call AWS.Request.send() on the returned request object to initiate the request.

  Context (`this`):

  - (`AWS.Response`) —

    the response object containing error, data properties, and the original request object.

  Parameters:

  - err (`Error`) —

    the error object returned from the request. Set to `null` if the request is successful.

  - data (`Object`) —

    the de-serialized data returned from the request. Set to `null` if a request error occurs. The `data` object has the following properties:

    - `Attributes` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `ConsumedCapacity` — (`map`)
      - `TableName` — (`String`)
      - `CapacityUnits` — (`Float`)
      - `Table` — (`map`)
        - `CapacityUnits` — (`Float`)
      - `LocalSecondaryIndexes` — (`map<map>`)

      - `CapacityUnits` — (`Float`)
    - `GlobalSecondaryIndexes` — (`map<map>`)
      - `CapacityUnits` — (`Float`)
  - `ItemCollectionMetrics` — (`map`)
    - `ItemCollectionKey` — (`map<map>`) — a serializable JavaScript object. For information about the supported types see the DynamoDB Data Model
    - `SizeEstimateRangeGB` — (`Array<Float>`)

Returns:

- (`AWS.Request`) —

  a handle to the operation request for subsequent event callback registration.

See Also:

- AWS.DynamoDB.updateItem()

Generated on Thu Jan 28 14:11:13 2016 by yard 0.8.7.6 (ruby-2.2.1).