For example, consider the index created by the following operation:

```
db.collection.createIndex(
   {
     content: "text",
     "users.comments": "text",
     "users.profiles": "text"
   },
   {
     name: "MyTextIndex"
   }
)
```

Then, to remove this text index, pass the name `"MyTextIndex"` to the `db.collection.dropIndex()` method, as in the following:

```
db.collection.dropIndex("MyTextIndex")
```

To get the names of the indexes, use the `db.collection.getIndexes()` method.

Control Search Results with Weights

This document describes how to create a `text` index with specified weights for results fields.

For a `text` index, the *weight* of an indexed field denotes the significance of the field relative to the other indexed fields in terms of the score. The score for a given word in a document is derived from the weighted sum of the frequency for each of the indexed fields in that document. See `$meta` operator for details on returning and sorting by text scores.

The default weight is 1 for the indexed fields. To adjust the weights for the indexed fields, include the `weights` option in the `db.collection.createIndex()` method.

> **Warning:** Choose the weights carefully in order to prevent the need to reindex.

A collection `blog` has the following documents:

```
{ _id: 1,
  content: "This morning I had a cup of coffee.",
  about: "beverage",
  keywords: [ "coffee" ]
}

{ _id: 2,
  content: "Who doesn't like cake?",
  about: "food",
  keywords: [ "cake", "food", "dessert" ]
}
```

To create a `text` index with different field weights for the `content` field and the `keywords` field, include the `weights` option to the `createIndex()` method. For example, the following command creates an index on three fields and assigns weights to two of the fields:

```
db.blog.createIndex(
                      {
                        content: "text",
                        keywords: "text",
                        about: "text"
                      },
                      {
```