Chapter 1: Basic Aggregation - $match and $project

Optional Lab - Expressions with $project

This lab will have you work with data within arrays, a common operation.

Specifically, one of the arrays you'll work with is `writers`, from the **movies** collection.

There are times when we want to make sure that the field is an array, and that it is not empty. We can do this within `$match`

```
{ $match: { writers: { $elemMatch: { $exists: true } } }
```

COPY

However, the entries within `writers` presents another problem. A good amount of entries in `writers` look something like the following, where the writer is attributed with their specific contribution

```
"writers" : [ "Vincenzo Cerami (story)", "Roberto Benigni (
```

COPY

But the writer also appears in the `cast` array as "Roberto Benigni"!

Give it a look with the following query

```
db.movies.findOne({title: "Life Is Beautiful"}, { _id: 0, c
```

COPY

This presents a problem, since comparing `"Roberto Benigni"` to `"Roberto Benigni (story)"` will definitely result in a difference.

Thankfully there is a powerful expression to help us, `$map`. `$map` lets us iterate over an array, element by element, performing some transformation on each element. The result of that transformation will be returned in the same place as the original element.

Within `$map`, the argument to `input` can be any expression as long as it resolves to an array. The argument to `as` is the name of the variable we want to use to refer to each element of the array when performing whatever logic we want. The field `as` is optional, and if omitted each element must be referred to as `$$this`:: The argument to `in` is the expression that is applied to each element of the `input` array, referenced with the variable name specified in `as`, and prepending two dollar signs:

COPY

```
writers: {
  $map: {
    input: "$writers",
    as: "writer",
    in: "$$writer"
```

`in` is where the work is performed. Here, we use the `$arrayElemAt` expression, which takes two arguments, the array and the index of the element we want. We use the `$split` expression, splitting the values on `" ("`.

If the string did not contain the pattern specified, the only modification is it is wrapped in an array, so `$arrayElemAt` will always work

COPY

```
writers: {
  $map: {
    input: "$writers",
    as: "writer",
    in: {
      $arrayElemAt: [
        {
          $split: [ "$$writer", " (" ]
        },
        0
      ]
    }
  }
}
```

Proceed to next section