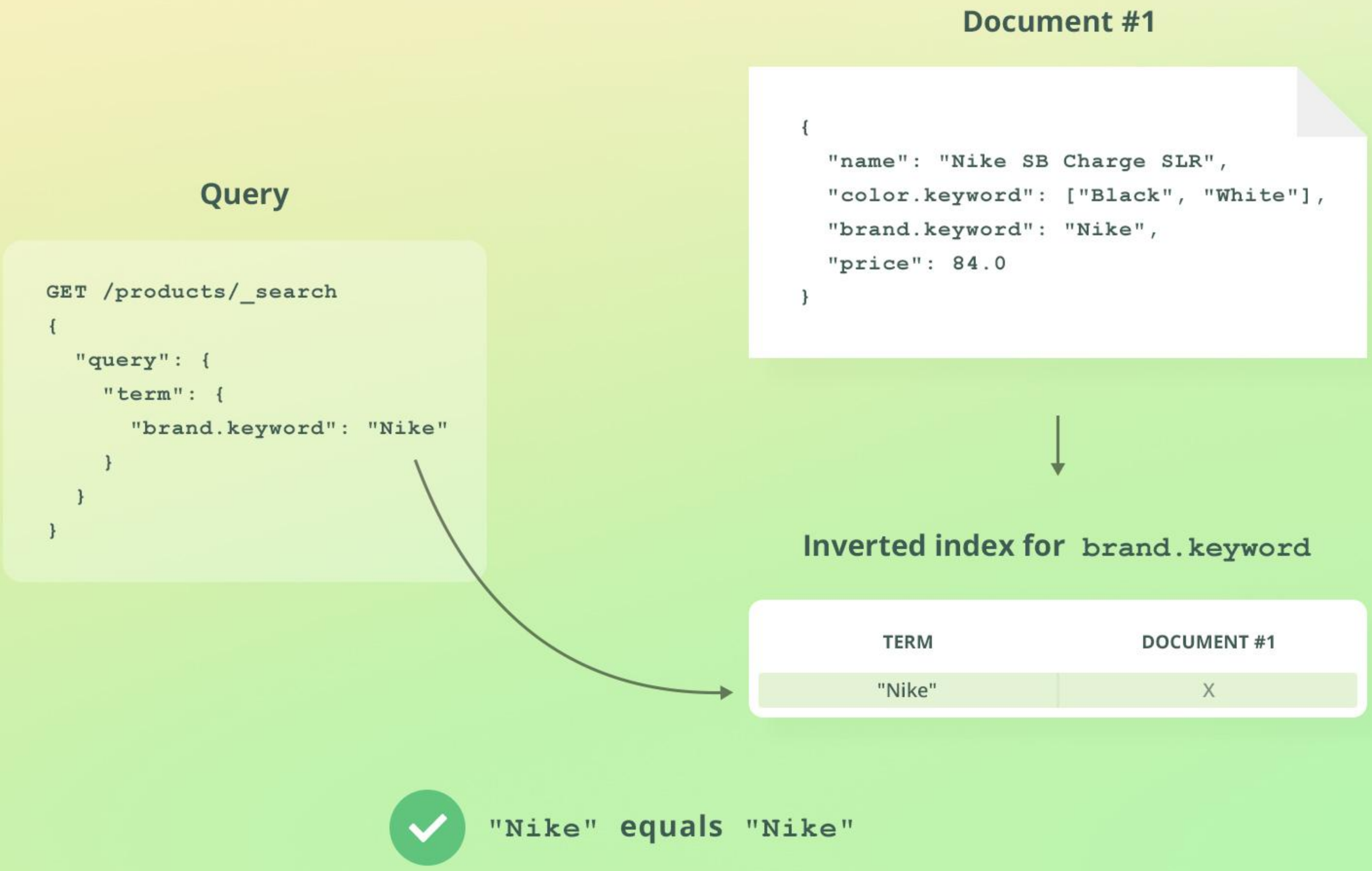
Introduction to term level queries

Introduction

- One group of Elasticsearch queries is called *term level queries*
- Used to search structured data for exact values (filtering)
    - E.g. finding products where the brand equals "Nike"
- **Term level queries are not analyzed**
    - The search value is used exactly as is for inverted index lookups
- Can be used with data types such as `keyword`, numbers, dates, etc.
    - Just **don’t** use them for `text` fields!



Query

```
GET /products/_search
{
  "query": {
    "term": {
      "brand.keyword": "Nike"
    }
  }
}
```

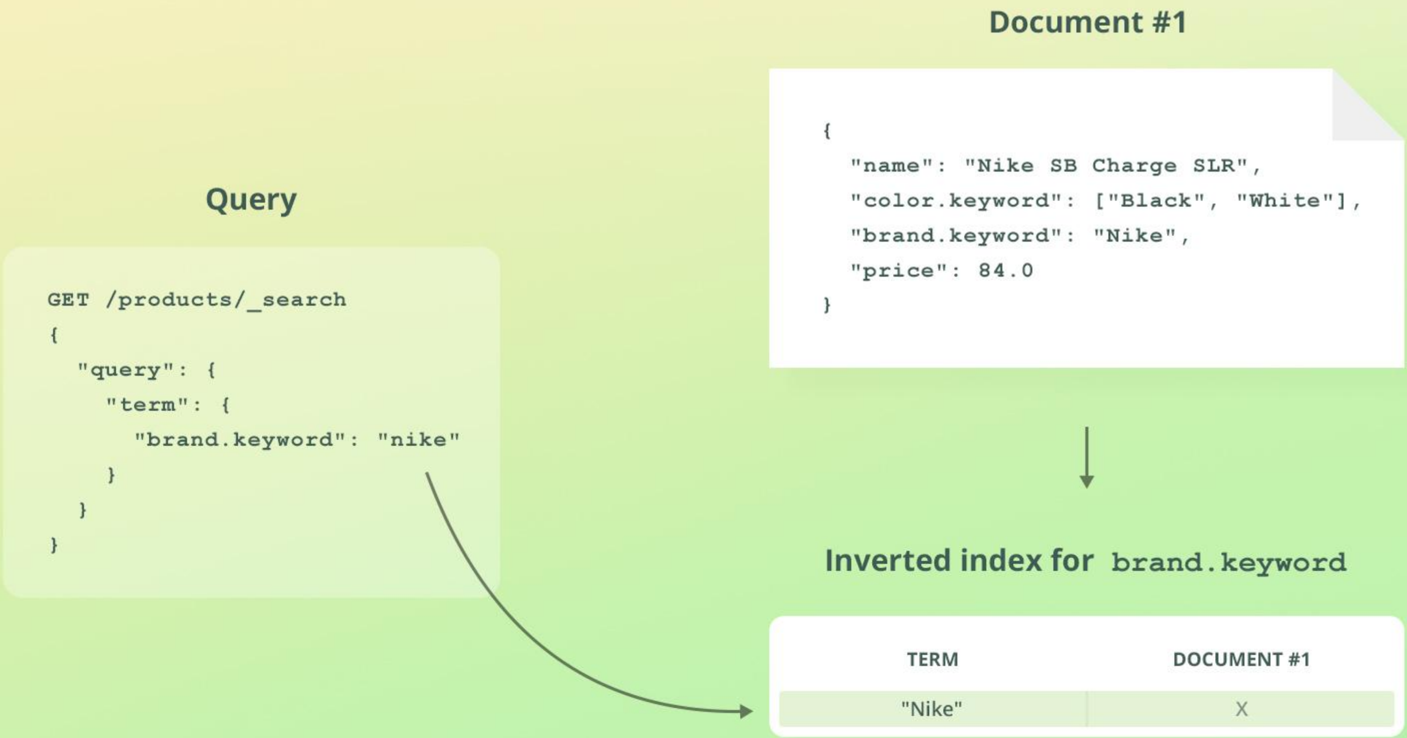
Document #1

```
{
  "name": "Nike SB Charge SLR",
  "color.keyword": ["Black", "White"],
  "brand.keyword": "Nike",
  "price": 84.0
}
```

Inverted index for `brand.keyword`

TERM	DOCUMENT #1
"Nike"	X

✓ `"Nike"` **equals** `"Nike"`



Document #1

```
{
  "name": "Nike SB Charge SLR",
  "color.keyword": ["Black", "White"],
  "brand.keyword": "Nike",
  "price": 84.0
}
```

Query

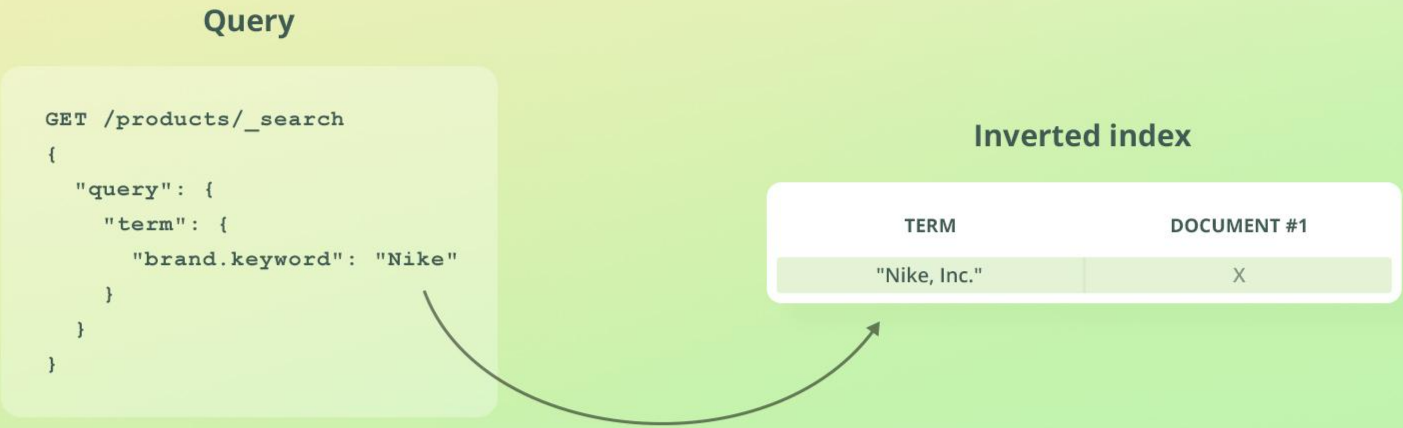
```
GET /products/_search
{
  "query": {
    "term": {
      "brand.keyword": "nike"
    }
  }
}
```

Inverted index for `brand.keyword`

TERM	DOCUMENT #1
"Nike"	X

✖ `"nike"` **does not equal** `"Nike"`

The entire value must match



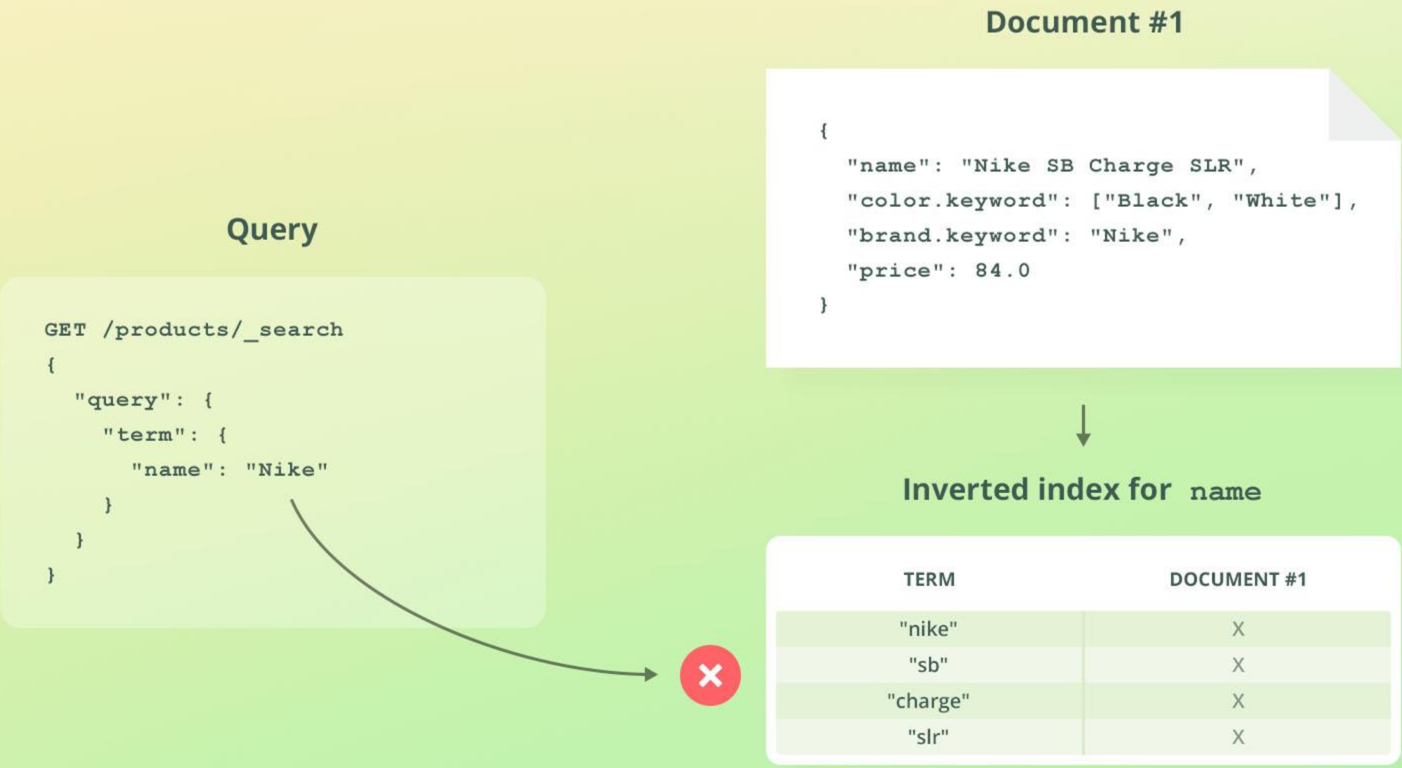
Query

```
GET /products/_search
{
  "query": {
    "term": {
      "brand.keyword": "Nike"
    }
  }
}
```

Inverted index

TERM	DOCUMENT #1
"Nike, Inc."	X

✖ `"Nike"` **does not equal** `"Nike, Inc."`



Query

```
GET /products/_search
{
  "query": {
    "term": {
      "name": "Nike"
    }
  }
}
```

Document #1

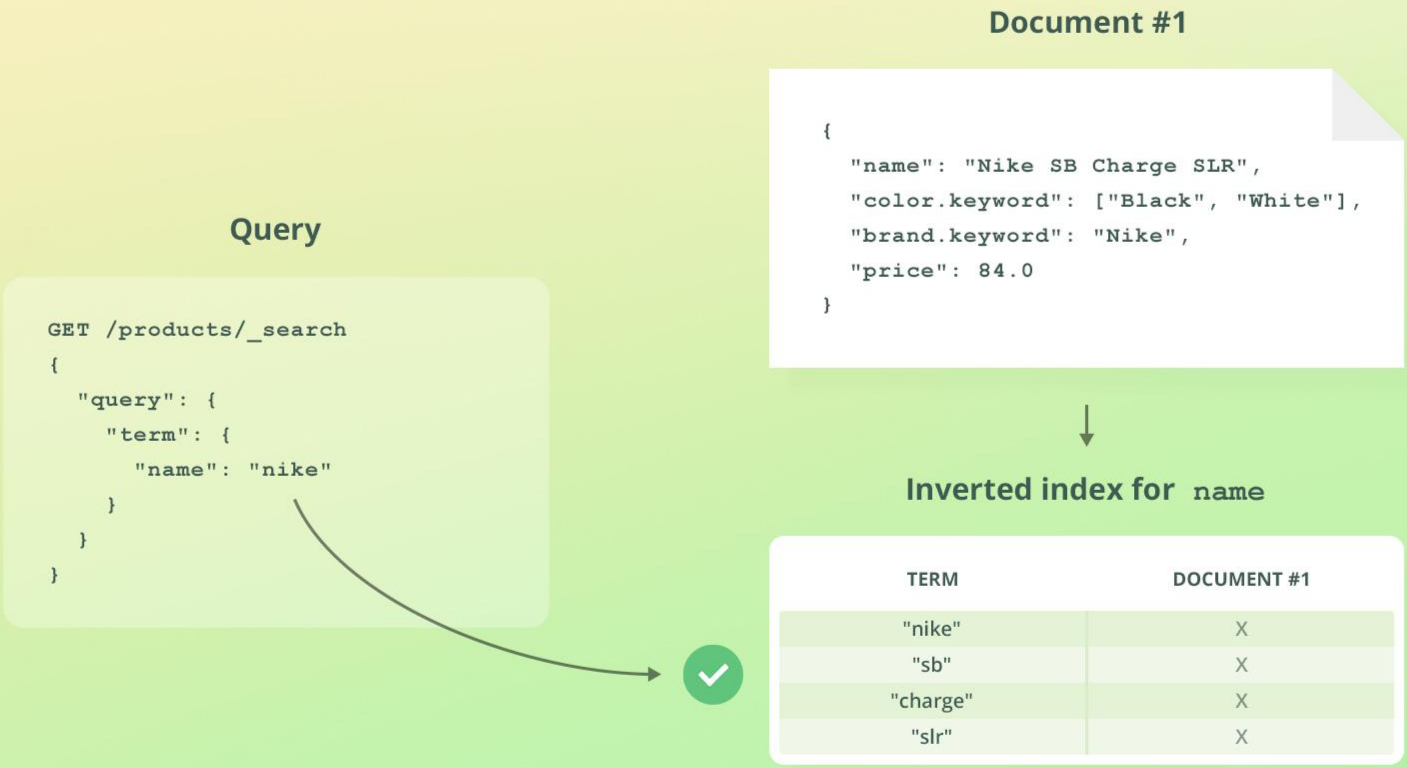
```
{
  "name": "Nike SB Charge SLR",
  "color.keyword": ["Black", "White"],
  "brand.keyword": "Nike",
  "price": 84.0
}
```

Inverted index for `name`

TERM	DOCUMENT #1
"nike"	X
"sb"	X
"charge"	X
"slr"	X

`"Nike"` **not found**

CodingExplained
CODE. SLEEP. REPEAT.



Query

```
GET /products/_search
{
  "query": {
    "term": {
      "name": "nike"
    }
  }
}
```

Document #1

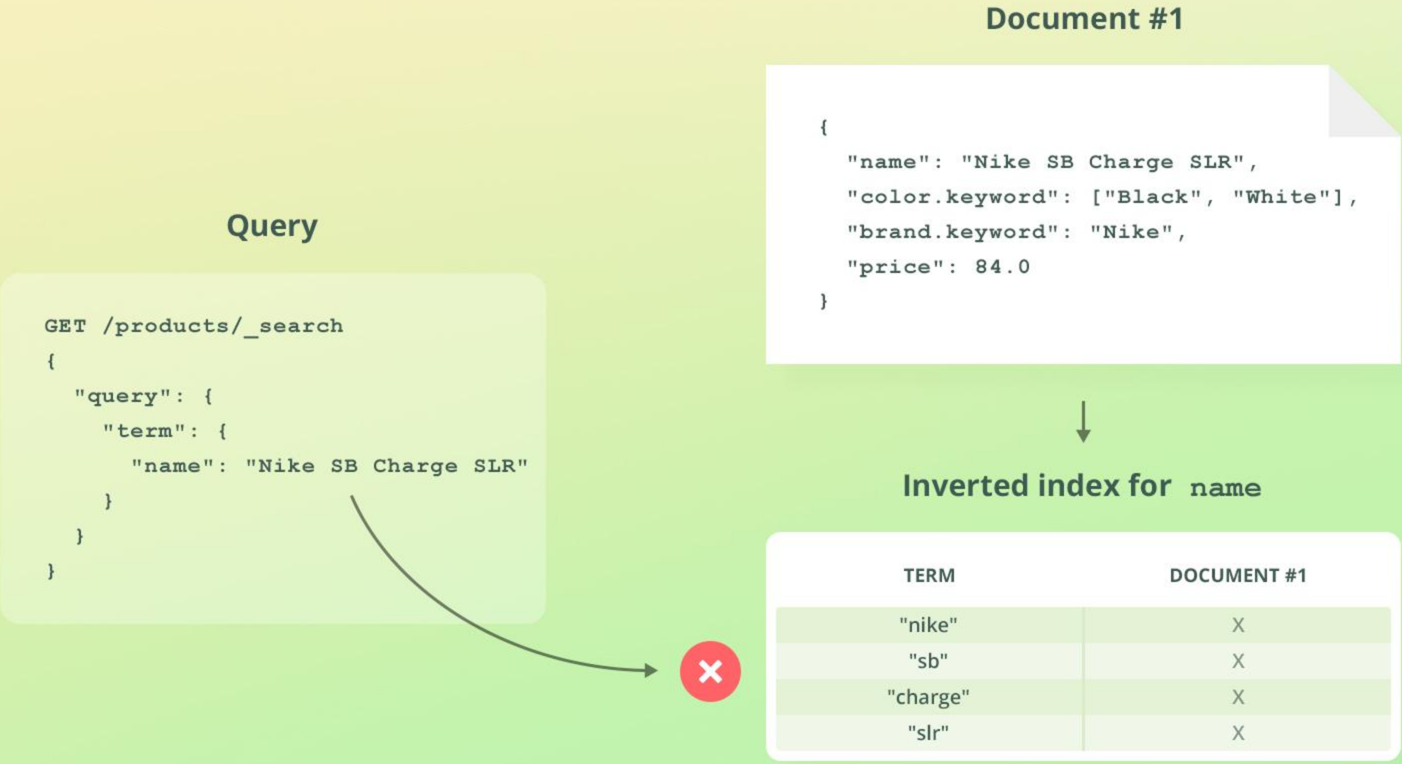
```
{
  "name": "Nike SB Charge SLR",
  "color.keyword": ["Black", "White"],
  "brand.keyword": "Nike",
  "price": 84.0
}
```

Inverted index for `name`

TERM	DOCUMENT #1
"nike"	X
"sb"	X
"charge"	X
"slr"	X

`"nike"` **equals** `"nike"`

CodingExplained
CODE. SLEEP. REPEAT.



Query

```
GET /products/_search
{
  "query": {
    "term": {
      "name": "Nike SB Charge SLR"
    }
  }
}
```

Document #1

```
{
  "name": "Nike SB Charge SLR",
  "color.keyword": ["Black", "White"],
  "brand.keyword": "Nike",
  "price": 84.0
}
```

Inverted index for `name`

TERM	DOCUMENT #1
"nike"	X
"sb"	X
"charge"	X
"slr"	X

The term is not found

So what’s the big deal?

- Results can seem strange and unpredictable (but of course they aren’t)
- *“It works; deploy!”*
- Queries don’t fail; they just don’t behave as expected
- A query for “nike” works fine, but “Nike” doesn’t match anything
- Can be hard to debug if you don’t understand how the query works
- *Many* people have run into this - don’t be the next one 😉
- **TL;DR:** Don’t use term level queries on `text` fields. That’s it!

Lecture summary

- Term level queries are used for exact matching (filtering)
- Term level queries are not analyzed
  - The search value is used exactly as is for inverted index lookups
  - Searches are therefore case sensitive
  - The entire term must match (`"Nike"` doesn't match `"Nike, Inc."`)
- Can be used with multiple data types
  - **Don't** use them for the `text` data type (!!!)