`standard` analyzer

- Splits text at word boundaries and removes punctuation
  - Done by the `standard` tokenizer
- Lowercases letters with the `lowercase` token filter
- Contains the `stop` token filter (disabled by default)

`standard` analyzer (example)

```
"Is that Peter's cute-looking dog?"
```

```
["is", "that", "peter's", "cute", "looking", "dog"]
```

`simple` analyzer

- Similar to the `standard` analyzer
  - Splits into tokens when encountering *anything else* than letters
- Lowercases letters with the `lowercase` tokenizer
  - Unusual and a performance hack

`simple` analyzer (example)

```
"Is that Peter's cute-looking dog?"
```

```
["is", "that", "peter", "s", "cute", "looking", "dog"]
```

`whitespace` analyzer

- Splits text into tokens by whitespace
- Does *not* lowercase letters

```
"Is that Peter's cute-looking dog?"
```

```
["Is", "that", "Peter's", "cute-looking", "dog?"]
```

`keyword` analyzer

- No-op analyzer that leaves the input text intact
  - It simply outputs it as a single token
- Used for `keyword` fields by default
  - Used for exact matching

`keyword` analyzer (example)

```
"Is that Peter's cute-looking dog?"
```

```
["Is that Peter's cute-looking dog?"]
```

`pattern` analyzer

- A regular expression is used to match token separators
  - It should match whatever should split the text into tokens
- This analyzer is very flexible
- The default pattern matches all non-word characters (`\W+`)
- Lowercases letters by default

`pattern` analyzer (default configuration)

```
"Is that Peter's cute-looking dog?"
```

```
["is", "that", "peter", "s", "cute", "looking", "dog"]
```

CodingExplained
CODE. SLEEP. REPEAT.