Voice Dev 2

CS571: Building User Interfaces

Cole Nelson

Today's Warmup

- Clone today's code to your machine.
  - Run the command `npm install` inside of the `starter` and `solution` folders.

Learning Objectives

1. Be able to define and extract entities from utterances.
2. Be able to fully grasp the asynchronous nature of JavaScript.
3. Be able to apply this understanding to existing codebases.

Last Time...

Let's create a Wit.AI comedian that can understand...

`why_chicken` e.g. "why did the chicken cross the road"

- We will reply with "to get to the other side!"

`tell_joke` e.g. "tell me a joke"

- We will reply with jokes fetched from the Jokes API!

```
const createChatAgent = () => {
    const CS571_WITAI_ACCESS_TOKEN = "...";

    // Define variables here!
    let jokeNum;

    const handleInitialize = async () => {
        return "Welcome to BadgerChat Jokes!";
    }

    // ...
}
export default createChatAgent;
```

Closures allow us to share some *state*. Should we be concerned about sharing this?

Asynchronous Code

❗ Two `async` functions sharing state? JavaScript is actually single-threaded!

❗❗ JavaScript is single-threaded? Things like `fetch` and `setTimeout` are run by the browser *outside of JavaScript* on a seperate thread.

Enter the **event loop**.

Event Loop

1. When an asynchronous function is invoked, like *fetch* or *setTimeout*, begin work outside of JavaScript.
2. When the work is complete add the callback function to the task queue.
3. When the stack is empty, pull and execute the next callback function from the task queue*.

Learn more about the event loop... (1) (2)

* since 2015, there is technically a task (js) *and* microtask (us) queue

Image Source, Slightly Modified

A Deeper Understanding

Let's deconstruct the React app surrounding our agent.

Entities

We can get all kinds of jokes...

https://v2.jokeapi.dev/joke/any?safe-mode

... or we can get specific jokes!

https://v2.jokeapi.dev/joke/spooky?safe-mode

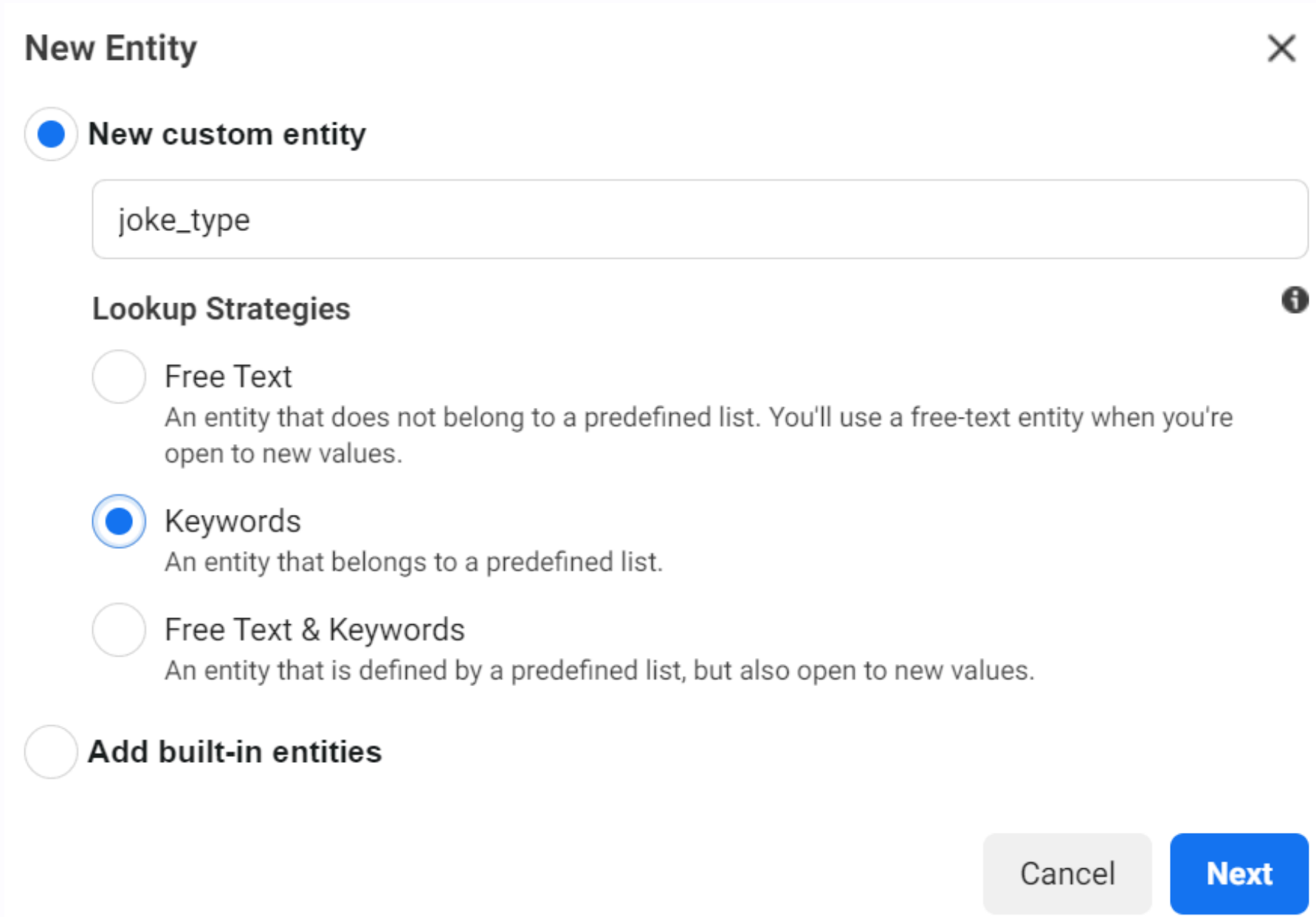
How can we handle this?

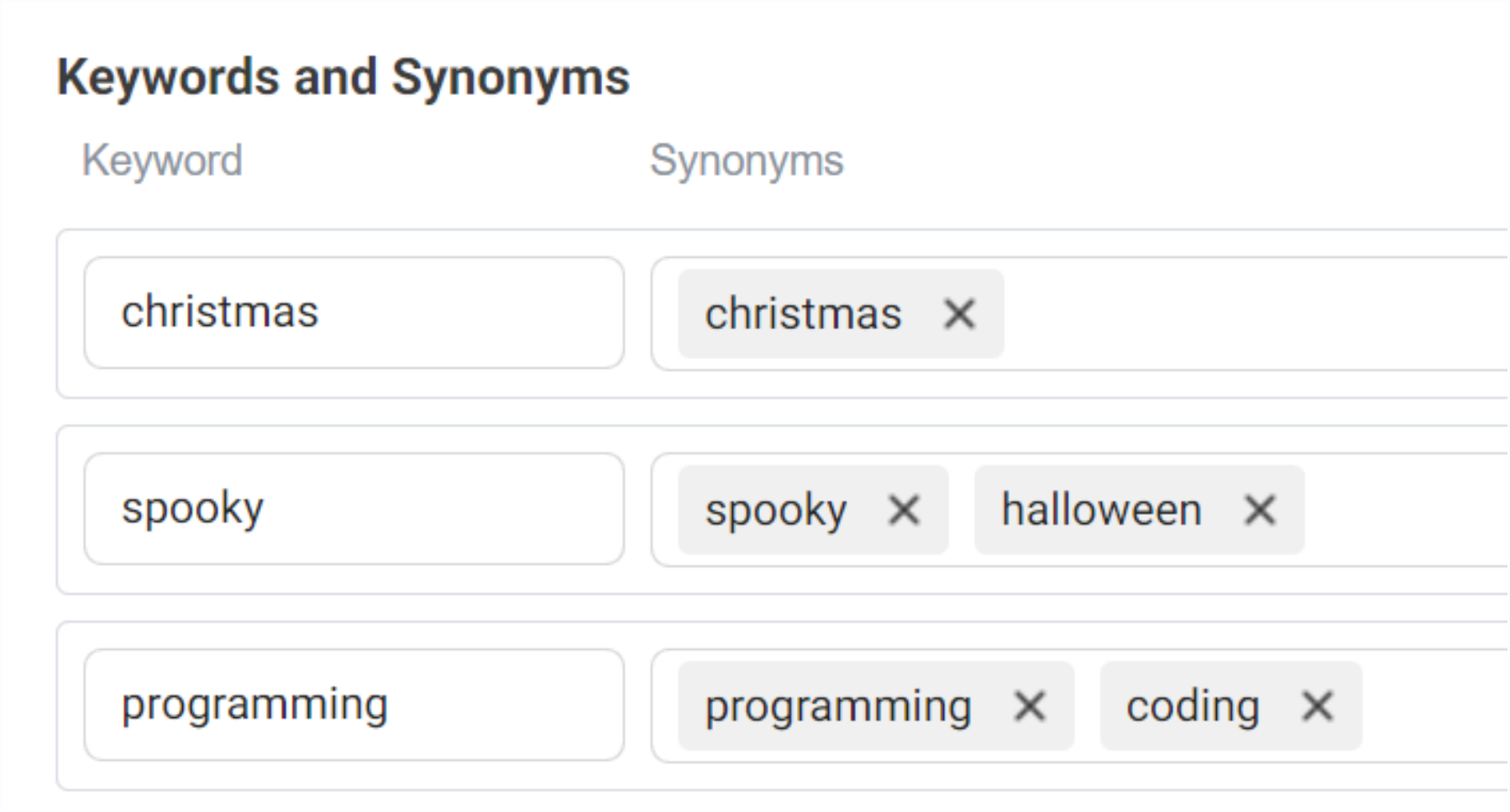
Entities

We could create the following intents...

- `tell_pun_joke`
- `tell_spooky_joke`
- `tell_programming_joke`
- ...

... or make an entity parameter, e.g. `joke_type`!


New Entity
New custom entity
joke_type
Lookup Strategies
Free Text
An entity that does not belong to a predefined list. You'll use a free-text entity when you're open to new values.
Keywords
An entity that belongs to a predefined list.
Free Text & Keywords
An entity that is defined by a predefined list, but also open to new values.
Add built-in entities
Cancel
Next


Keywords and Synonyms
Keyword
Synonyms
christmas
christmas
spooky
spooky
halloween
programming
programming
coding

```
{
    "entities": {
        "joke_type:joke_type": [
            {
                "body": "halloween",
                "confidence": 1,
                "end": 19,
                "entities": {},
                "id": "948991563175475",
                "name": "joke_type",
                "role": "joke_type",
                "start": 10,
                "type": "value",
                "value": "spooky"
            }
        ]
    },
    "intents": [
        {
            "confidence": 0.992232100272184,
            "id": "2117974348567211",
            "name": "tell_joke"
        }
    ],
    "text": "tell me a halloween joke",
    "traits": {}
}
```

Intents & Entities

The JSON response body consists of...

- `text` - exactly what the user said
- `intents` - a *list* of likely matches
- `entities` - an *object* of likely matches
- `traits` - maybe next time? 😊😔

Read the Wit.AI Docs

Intents

```
"intents": [
    {
        "confidence": 0.992232100272184,
        "id": "2117974348567211",
        "name": "tell_joke"
    }
]
```

Each intent consists of a `confidence` from 0 to 1, as well as an `id` linked to the `name` of the intent.

The 0th intent is the best match.

Entities

```
"entities": {
    "joke_type:joke_type": [
        {
            "body": "halloween",
            ...
            "value": "spooky"
        }
    ]
}
```

Entities map a specific type to a list of `body` (what was actually written) and `value` (what it was resolved to)

Your Turn!

Let's build a better comedian.

Questions?