How to code a Telegram bot with Jennifer



A complete interactive bot in one short program: commands, buttons, photos, flood control, and a clean stop, with the `@jennifer/telegram` deck.

Based on **Jennifer 0.26.0**, November 2026, by @mplx.

Long polling: why it fits in one program

- The bot **asks** Telegram for updates: `getUpdates`, waiting up to 25 s
- No public HTTPS endpoint, no certificate, no open port: runs behind NAT
- Nothing to verify: the bot talks only to Telegram, over HTTPS
- One loop, one process; the state is a single number, the **offset**

The alternative, a webhook, is on the last slide.

Set up

1. In Telegram, talk to @**BotFather**: `/newbot`, a name, a username
2. It answers with a token like `123456:ABC-...`: a password, treat it as one
3. Add the deck:

```
jvc add @jennifer/telegram     # into vendor/jennifer/telegram/
export TELEGRAM_BOT_TOKEN=...  # from a secret store, never in git

import "@jennifer/telegram/" as telegram;
```

First contact: getMe

```
def token as string init os.getEnv("TELEGRAM_BOT_TOKEN");
if ($token == "") {
    io.eprintf("bot: set TELEGRAM_BOT_TOKEN\n");
    exit 2;
}
def b as telegram.Bot init telegram.bot($token);
def me as telegram.User init telegram.getMe($b);
io.printf("running as @%s\n", $me.username);
```

```
running as @jennifer_dice_bot
```

The loop and the offset

```
try {                                       # in the while loop
    $updates = telegram.getUpdatesRaw($b, $offset, POLL_SECONDS);
} catch (e) {
    def why as string init telegram.redactToken($e.message, $token);
    io.printf("poll failed (%s): %s\n", $e.kind, $why);
    time.sleep(time.fromSeconds(5));
}
for (def raw in $updates) {
    $offset = telegram.parseUpdate($raw).updateId + 1;
    handle($b, $raw);    # try { onUpdate(...) } catch: log, go on
}
```

What Telegram saw

```
tg: getUpdates offset=0 timeout=25 -> 3 update(s)
tg: sendMessage chat=42 +keyboard "Hi! Pick a roll:"
tg: answerCallbackQuery cb2
tg: sendMessage chat=42 "2d6: 4 + 5 = 9"
tg: sendMessage chat=42 "3d20: 13 + 20 + 2 = 35"
tg: getUpdates offset=4 timeout=25 -> 3 update(s)
tg: sendMessage chat=42 "1 to 20 dice, 2 to 1000 sides."
tg: sendChatAction upload_photo
tg: sendPhoto multipart, 5894 bytes
tg: getUpdates offset=7 timeout=25 -> 0 update(s)
```

Commands: match on the first word

```
def cmd as string init strings.split($words[0], "@")[0];   # /roll@bot
def id as int init $msg.chatId;
match ($cmd) {
    when "/start" {
        telegram.sendMessageWithKeyboard($b, $id, START, keyboard());
    }
    when "/help" { reply($b, $id, HELP); }
    when "/roll" { reply($b, $id, roll($arg)); }   # 2nd word or 1d6
    when "/logo" { sendLogo($b, $id); }
    else { reply($b, $id, HELP); }
}
```

/roll: never trust the input

```
if (not regex.matches('^[0-9]{1,4}d[0-9]{1,4}$', $spec)) {
    return "Write it like /roll 2d6.";
}
def parts as list of string init strings.split($spec, "d");
def n as int init convert.toInt($parts[0]);
def sides as int init convert.toInt($parts[1]);
if ($n < 1 or $n > 20 or $sides < 2 or $sides > 1000) {
    return "1 to 20 dice, 2 to 1000 sides.";
}
```

```
<- 3 /roll 3d20        ->  3d20: 13 + 20 + 2 = 35
<- 4 /roll 900d6       ->  1 to 20 dice, 2 to 1000 sides.
```

Buttons and callback queries

```
[[telegram.inlineButton("Roll 1d6", "roll:1d6"),
  telegram.inlineButton("Roll 2d6", "roll:2d6")],
 [telegram.urlButton("Jennifer", "https://jennifer-lang.dev/")]]

def q as telegram.CallbackQuery init telegram.parseCallbackQuery($raw);
telegram.answerCallbackQuery($b, $q.id, "");
# CallbackQuery has no chat id; read it from the raw update
def chat as int init json.asInt($raw, CHAT_ID);   # its JSON Pointer
if (strings.startsWith($q.data, "roll:")) {
    reply($b, $chat, roll($q.data[5..]));
}
```

Sending a photo or a file

```
func sendLogo(b as telegram.Bot, chatId as int) {
    telegram.sendChatAction($b, $chatId, "upload_photo");
    telegram.sendPhotoFile($b, $chatId, "jennifer-logo.png", "image/
png");
}

<- 5 /logo
tg: sendChatAction upload_photo
tg: sendPhoto multipart, 5894 bytes
```

- `sendPhoto(b, chat, urlOrFileId, caption)`: no upload, Telegram fetches it
- `sendDocumentFile(b, chat, path, contentType)`: any file, as a document

Errors: a struct, flood control, no token in logs

```
try {                                   # in reply(), up to three times
    return telegram.sendMessage($b, $chatId, $text);
} catch (e) {
    if ($e.kind != "telegram" or $e.retryAfter == 0) { throw $e; }
    io.printf("flood control: wait %d s\n", $e.retryAfter);
    time.sleep(time.fromSeconds($e.retryAfter));
}

<- 6 text
tg: sendMessage -> 429
flood control: wait 2 s
tg: sendMessage chat=42 "I know /start, /help, /roll 2d6 and /logo."
```

`TelegramError`: `errorCode`, `retryAfter`, `migrateToChatId`, `status`

Run it as a service, stop it cleanly

```
[Service]
ExecStart=/usr/bin/jennifer run /opt/dicebot/bot.j
WorkingDirectory=/opt/dicebot
EnvironmentFile=/etc/dicebot.env
DynamicUser=yes
Restart=on-failure
```

```
os.catchSignal("term");           # before the loop
if (os.gotSignal("term") or os.gotSignal("int")) { $running = false; }
telegram.getUpdatesRaw($b, $offset, 0);   # after it, in a try
```

Test without Telegram

```
def b as telegram.Bot init telegram.bot($token);
if (os.getEnv("TELEGRAM_API") != "") {
    $b = telegram.botWith($token, os.getEnv("TELEGRAM_API"));
}
```

```
$ jennifer run fake-telegram.j 28780 &
$ TELEGRAM_BOT_TOKEN=123456:TEST-TOKEN \
  TELEGRAM_API=http://127.0.0.1:28780 jennifer run bot.j
running as @jennifer_dice_bot
```

A fake Bot API in 134 lines: `httpd`, scripted updates, one 429.

Webhooks, gaps, and read more

- **Webhook instead of polling:** when the bot already sits behind a public HTTPS server or must scale to many messages. The deck does not register webhooks; `telegram.parseUpdate` reads the same JSON in a `web` handler.
- **Not yet in the deck:** the chat id of a button press, editing a sent message, the sender of a message (all reachable through the raw JSON)
- The deck: jennifer-language/deck-telegram
- The language: jennifer-lang.dev, this talk in the Jennifer Academy

Based on Jennifer 0.26.0, November 2026.