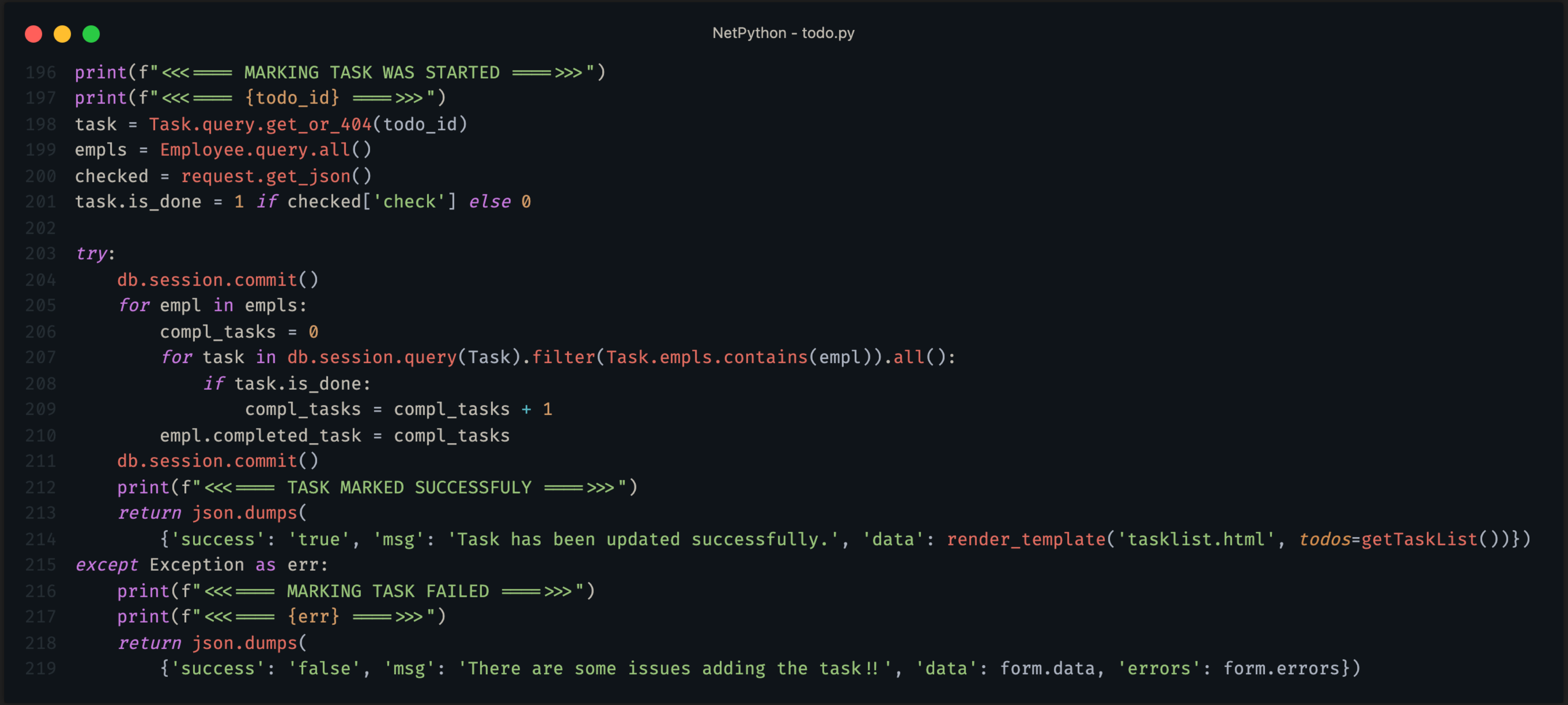
How to use Ajax with Flask

What is Ajax

- Ajax (Asynchronous JavaScript and XML) is a set of web development techniques using many web technologies on the client-side to create asynchronous web applications.
- With Ajax, web applications can send a retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

So, how to do this…

Let’s see at flask route



NetPython - todo.py

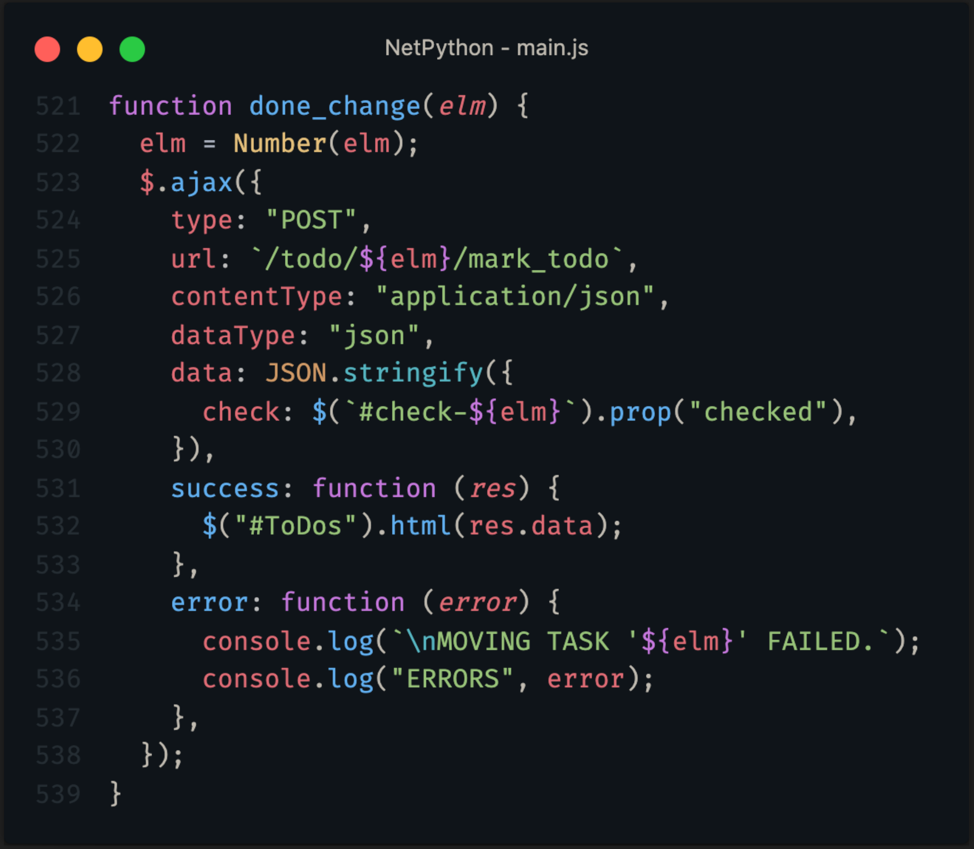
```python
print(f"<<<=== MARKING TASK WAS STARTED ===>>>")
print(f"<<<=== {todo_id} ===>>>")
task = Task.query.get_or_404(todo_id)
empls = Employee.query.all()
checked = request.get_json()
task.is_done = 1 if checked['check'] else 0

try:
    db.session.commit()
    for empl in empls:
        compl_tasks = 0
        for task in db.session.query(Task).filter(Task.empls.contains(empl)).all():
            if task.is_done:
                compl_tasks = compl_tasks + 1
        empl.completed_task = compl_tasks
    db.session.commit()
    print(f"<<<=== TASK MARKED SUCCESSFULY ===>>>")
    return json.dumps(
        {'success': 'true', 'msg': 'Task has been updated successfully.', 'data': render_template('tasklist.html', todos=getTaskList())})
except Exception as err:
    print(f"<<<=== MARKING TASK FAILED ===>>>")
    print(f"<<<=== {err} ===>>>")
    return json.dumps(
        {'success': 'false', 'msg': 'There are some issues adding the task!!', 'data': form.data, 'errors': form.errors})
```

In our case it is route for check task in todo list as done.
The main thing what we gonna do is return data as JSON. The reasons why we should send this data format:

- It’s very simple to use.
- We can easily check success status.

Ajax implementation



NetPython - main.js

```javascript
function done_change(elm) {
  elm = Number(elm);
  $.ajax({
    type: "POST",
    url: `/todo/${elm}/mark_todo`,
    contentType: "application/json",
    dataType: "json",
    data: JSON.stringify({
      check: $(`#check-${elm}`).prop("checked"),
    }),
    success: function (res) {
      $("#ToDos").html(res.data);
    },
    error: function (error) {
      console.log(`\nMOVING TASK '${elm}' FAILED.`);
      console.log("ERRORS", error);
    },
  });
}
```

And finally we come to Ajax implementation.
So the main, let's say, attributes of Ajax settings is: TYPE, URL and DATA.
TYPE – in WHICH method we send HTTP requests.
URL – address(in our case "route") WHERE we send these requests.
DATA – WHAT we send to server.

Then we should implement "success" function to get response from the server. And as for "error" function, it is optional. Cause it's get DB errors in most cases.

So, when we get success response we can use data to show what we did. In my case I receive "render_template" to write html file in "#ToDos" to update this div without refreshing the page.