Introduction to Web Development

Part 3: Demo NodeJs Rest Api

arnaud.nauwynck@gmail.com

Course Esilv 2023

This document:
https://github.com/Arnaud-Nauwynck/presentations/web/angular-demos-3-nodejs-rest-api.pdf

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

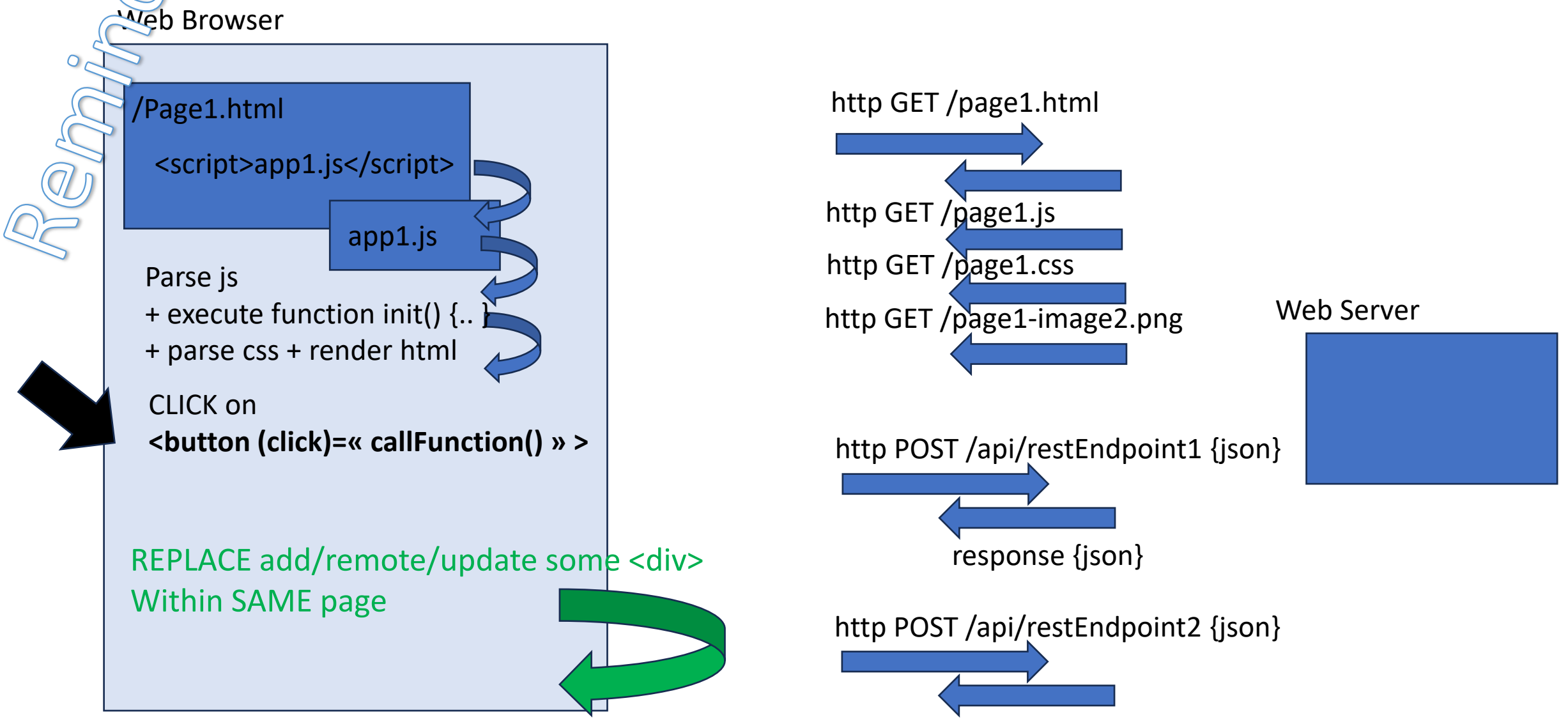
CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman
OpenApi – Swagger UI

SPA = Single Page Application
Single GET {html|js}, Multiple Rest Json requests

Reminder



Reminder

Rest using NodeJs - Express

```
const express = require('express');
const app = express();

app.get('/', function (req, res) {
  res.send({ some: 'Hello Json Express' })
});

app.listen(3000);
```

Reminder

~ Rest using Springboot: @RequestMapping

```
http PUT  /api/endpoint1/meth2
header1:value1

{
 « reqField »: « value»
}
```

```
http 200 OK
header2:value2

{
 « respField »: « value»
}
```

```
class RequestDTO {
  public String reqField;
}
```

Json -> java

Java -> json

```
class ResponseDTO {
  public String respField;
}
```

```
@RestController
@RequestMapping(« /api/endpoint1 »)
public class MyRestController {

    @PutMapping(« meth2»)
    public ResponseDTO handle(
          @RequestBody RequestDTO req) {
       return new …
    }
}
```

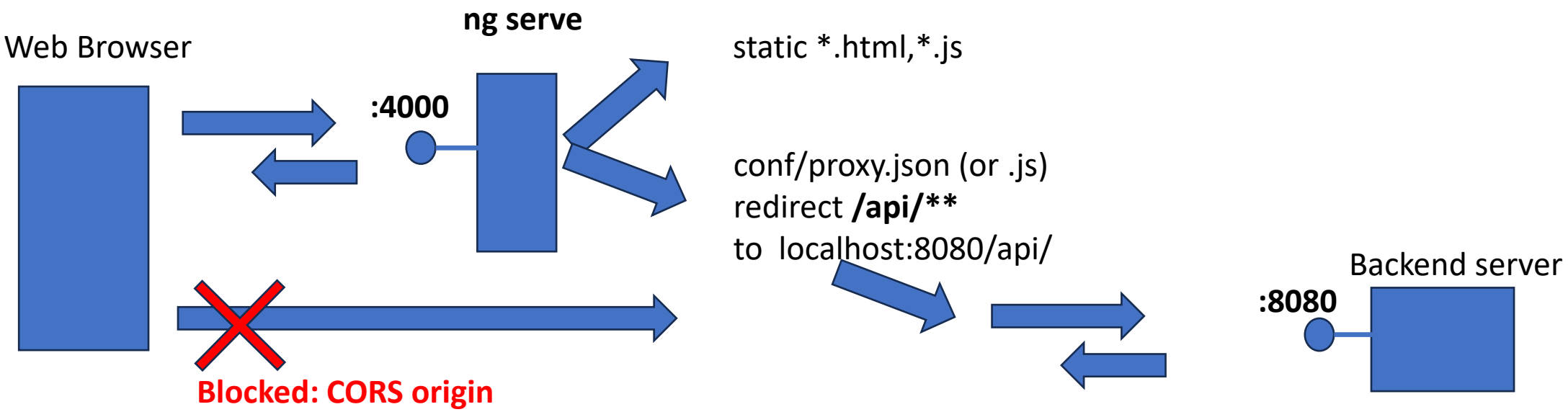
Reminder

Ng serve… why Proxy to /api/** ?
=> CORS Origin problem !

Because …

Web Browser blocks calls to « http://localhost:**8080**/api/** »

not comming from same « origin domain » as « http://localhost:**4000**/index.html » «app.js » !



Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

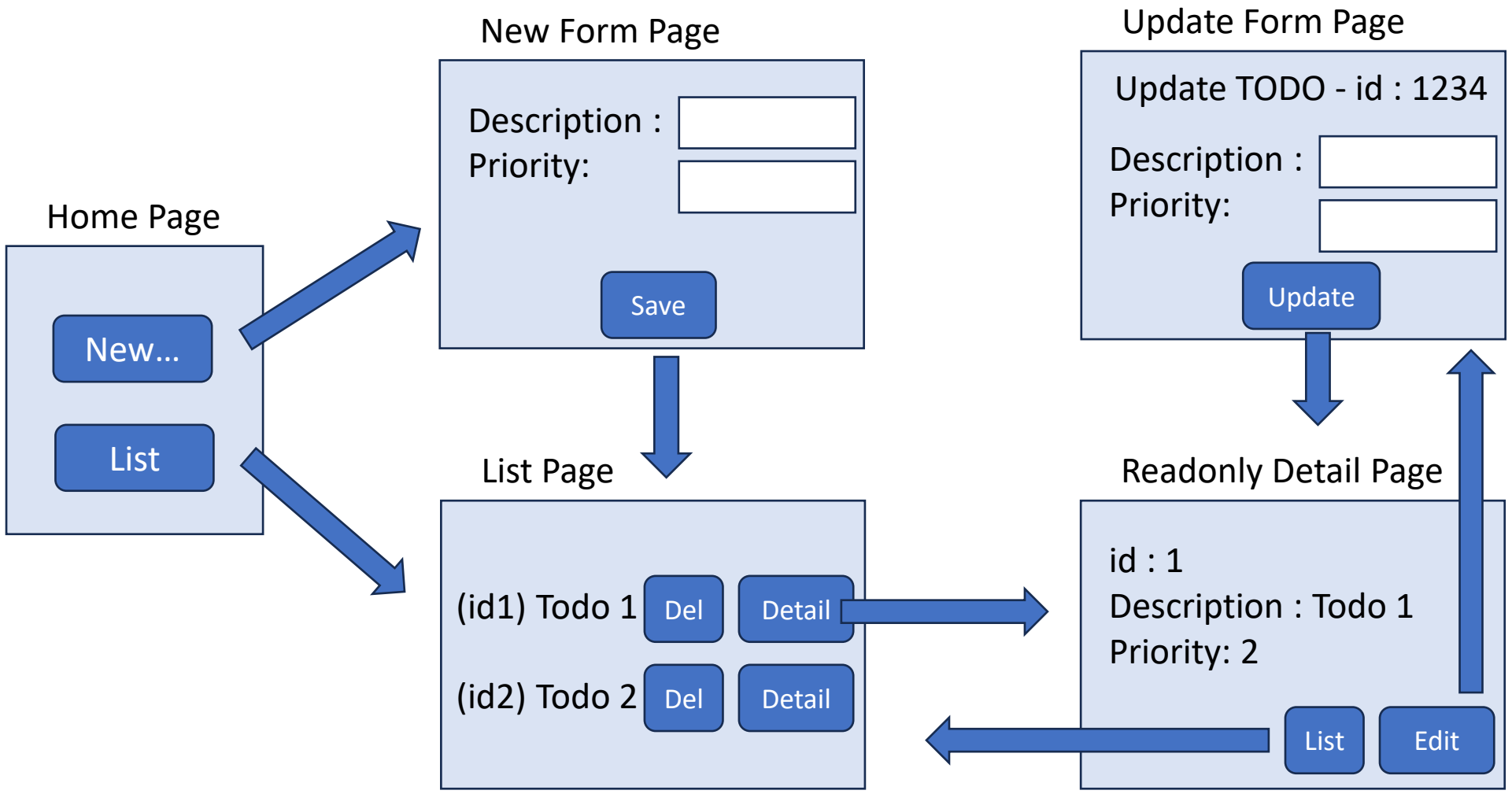
- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

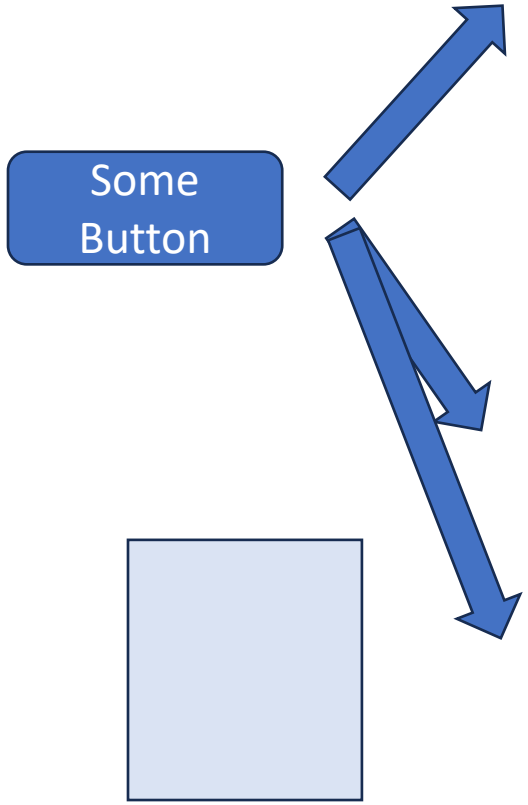
Test using Postman

OpenApi – Swagger UI

Web Site Sketch -> Discover API requirement



App Skeleton Buttons : Action / Query - Navigation



Button type Action => Execute Action on Server
= http POST, PUT, DELETE

Button type Query / Refresh => Execute Query on Server
= http GET

Button type Navigation => load page data = execute query
= http GET

Todo Web App API Description … CRUD

C **Create** ➡ Page 1: Simple Html Form with input fields
Click on « Save » button => save « Todo » object on server

R **Read All** ➡ Page 2: Show list of « Todos »

Read By Id ➡ page 3 for url « /todo/{id} »
Show detailed « Todo» page

U **Update** ➡ Page 4: Editable page + « update » button

D **Delete** ➡ « delete » button

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman
OpenApi – Swagger UI

$ npm init

```
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (demo-node-express-todo-rest)
version: (1.0.0)
description: rest server for Todos, using express
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\arn\perso\cours\esilv\web\demo-node-express-todo-rest\package.json:

{
  "name": "demo-node-express-todo-rest",
  "version": "1.0.0",
  "description": "rest server for Todos, using express",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

$ npm install –s express

```
$ npm install -s express
```

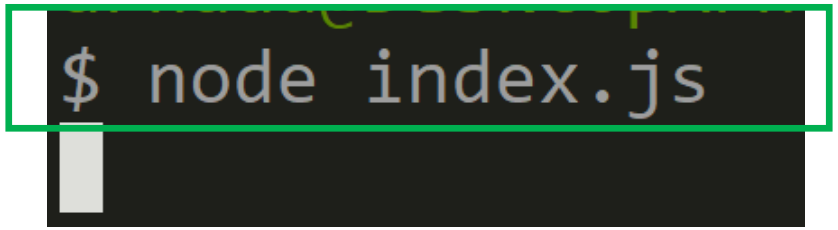
```
$ cat package.json
{
  "name": "demo-node-express-todo-rest",
  "version": "1.0.0",
  "description": "rest server for Todos, using express",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

Hello express endpoint (port 3000)

index.js

```
1  const express = require('express');
2  const app = express();
3
4  app.get('/', function (req, res) {
5    res.send({ some: 'Hello Json Express' })
6  });
7
8  app.listen(3000);
```

Running: $ node index.js



Test using Curl

```
$ curl -v http://localhost:3000
*   Trying 127.0.0.1:3000...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET / HTTP/1.1
> Host: localhost:3000
> User-Agent: curl/8.0.1
> Accept: */*
>
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 29
< ETag: W/"1d-qhxeNQT6M+LyU0MKyv0e0/zWvbE"
< Date: Mon, 31 Jul 2023 11:42:47 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
{"some":"Hello Json Express"}* Connection #0 to host localhost left intact
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman
OpenApi – Swagger UI

Rest endpoint : GET « /todo »

```
8   var todos = [
9       { id: 1, description: "learn Angular", priority: 1},
10      { id: 2, description: "learn Typescript", priority: 1},
11      { id: 3, description: "learn http", priority: 1},
12  ];
```

```
23  app.get('/todo', function (req, res) {
24    res.send(todos)
25  });
```

Test GET /todo using curl

```
$ curl http://localhost:3000/todo
[{"id":1,"description":"learn Angular","priority":1},{"id":2,"description":"learn Typescript","p
riority":1},{"id":3,"description":"learn http","priority":1}]
```

```
$ curl -v http://localhost:3000/todo
*   Trying 127.0.0.1:3000...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET /todo HTTP/1.1
> Host: localhost:3000
> User-Agent: curl/8.0.1
> Accept: */*
>
< HTTP/1.1 200 OK
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 157
< ETag: W/"9d-UoeCn1N4guaOZh3QvlvJpsfmzPY"
< Date: Mon, 31 Jul 2023 12:04:09 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
<
[{"id":1,"description":"learn Angular","priority":1},{"id":2,"description":"learn Typescript","p
riority":1},{"id":3,"description":"learn http","priority":1}]* Connection #0 to host localhost l
eft intact
```

Test GET /todo, using curl --silent + jq '.'

```
$ curl --silent http://localhost:3000/todo | jq '.'
[
  {
    "id": 1,
    "description": "learn Angular",
    "priority": 1
  },
  {
    "id": 2,
    "description": "learn Typescript",
    "priority": 1
  },
  {
    "id": 3,
    "description": "learn http",
    "priority": 1
  }
]
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman
OpenApi – Swagger UI

Rest endpoint POST /todo

```
14  var idGenerator = 4;
15  function generateNewId() {
16      return idGenerator++;
17  }
```

```
43  app.post('/todo', function (req, res) {
44    console.log('handle http POST /todo, request.body:', req.body);
45    const newId = generateNewId();
46    const todo = {id: newId, description: req.body.description, priority: req.body.priority };
47    todos.push(todo);
48    res.send(todo);
49  });
```

Add express support for json body-parser !!

index.js

```
1  const express = require('express');
2  const bodyParser = require('body-parser')
3
4  var app = express()
5
6  // parse application/json
7  app.use(bodyParser.json())
8
```

Test using curl …
-H « Content-Type: application/json »
-H « accept: application/json »
-d ’{« description»: « ..» }’

```
$ curl -X POST http://localhost:3000/todo -H "content-type:application/json" -H "accept: application/js
on" -d '{"description":"learn express", "priority":1}'
{"id":4,"description":"learn express","priority":1}
```

console.log() in nodejs

```
43  app.post('/todo', function (req, res) {
44    console.log('handle http POST /todo, request.body:', req.body);
```

```
$ node index.js
handle http POST /todo, request.body: { description: 'learn express', priority: 1 }
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman
OpenApi – Swagger UI

Rest endpoint GET « /todo/:id »

```
63  app.get('/todo/:id', function (req, res) {
64    const id = +req.params.id;
65    console.log('handle http GET /todo/:id, ', id);
66    const idx = todos.findIndex(x => x.id === id);
67    if (idx !== -1) {
68      res.send(todos[idx]);
69    } else {
70      res.status(404).send('Todo entity not found for id:' + id);
71    }
72  });
```

Notice… type coercion, exact === vs ==

```
const id = +req.param.id;   // <= coercion string to number
// equivalent to:
const idText = req.param.id; // idText is string
const id = +idText; // id is number

(id === 2)   // correct, exact equality for number
(id === ‘2’)  // wrong… types differ
(id == ‘2’)  or  (id == 2);  // both ok, using implicit type coercions
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman
OpenApi – Swagger UI

Rest endpoint PUT « /todo » req.body={ id:.., ..}

```
74  app.put('/todo', function (req, res) {
75    const reqBody = req.body;
76    console.log('handle http PUT /todo, request.body:', reqBody);
77    const id = reqBody.id;
78    const idx = todos.findIndex(x => x.id === id);
79    if (idx !== -1) {
80      const todo = todos[idx];
81      todo.description = reqBody.description;
82      todo.priority = reqBody.priority;
83      res.send(todo);
84    } else {
85      res.status(404).send('Todo entity not found for id:' + id);
86    }
87  });
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman
OpenApi – Swagger UI

Endpoint DELETE « /todo/:id »

```
51  app.delete('/todo/:id', function (req, res) {
52    const id = +req.params.id;
53    console.log('handle http DELETE /todo/:id, ', id);
54    const idx = todos.findIndex(x => x.id === id);
55    if (idx !== -1) {
56      const removed = todos.splice(idx, 1);
57      res.send(removed);
58    } else {
59      res.status(404).send('Todo entity not found for id:' + id);
60    }
61  });
```

Test Endpoint DELETE « /todo/:id »

```
$ curl -X DELETE http://localhost:3000/todo/2
[{"id":2,"description":"learn Typescript","priority":1}]
```

```
$ node index.js
handle http DELETE /todo/:id,  2
```

```
$ curl -X DELETE http://localhost:3000/todo/999999
Todo entity not found for id:999999
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

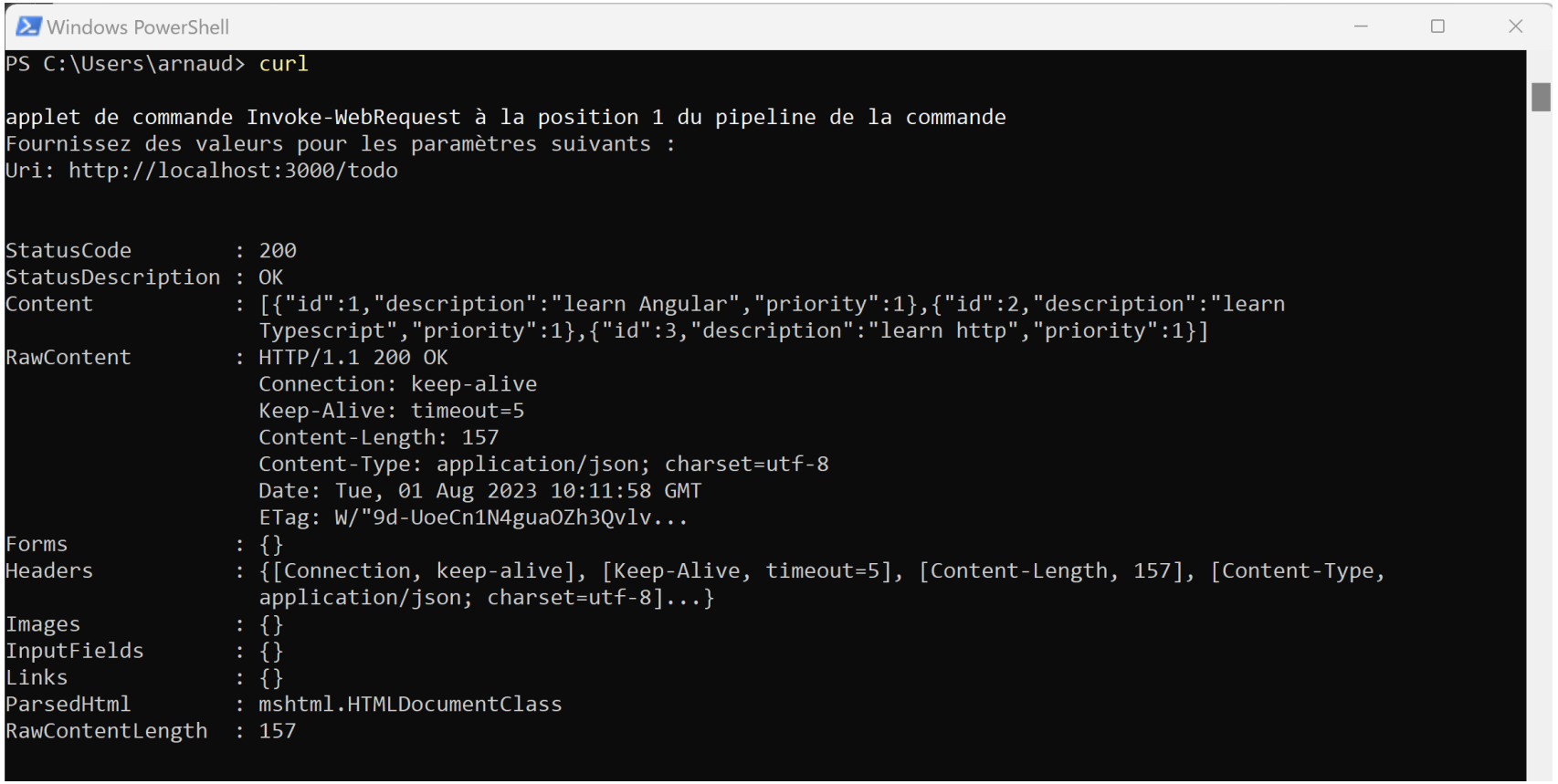
CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

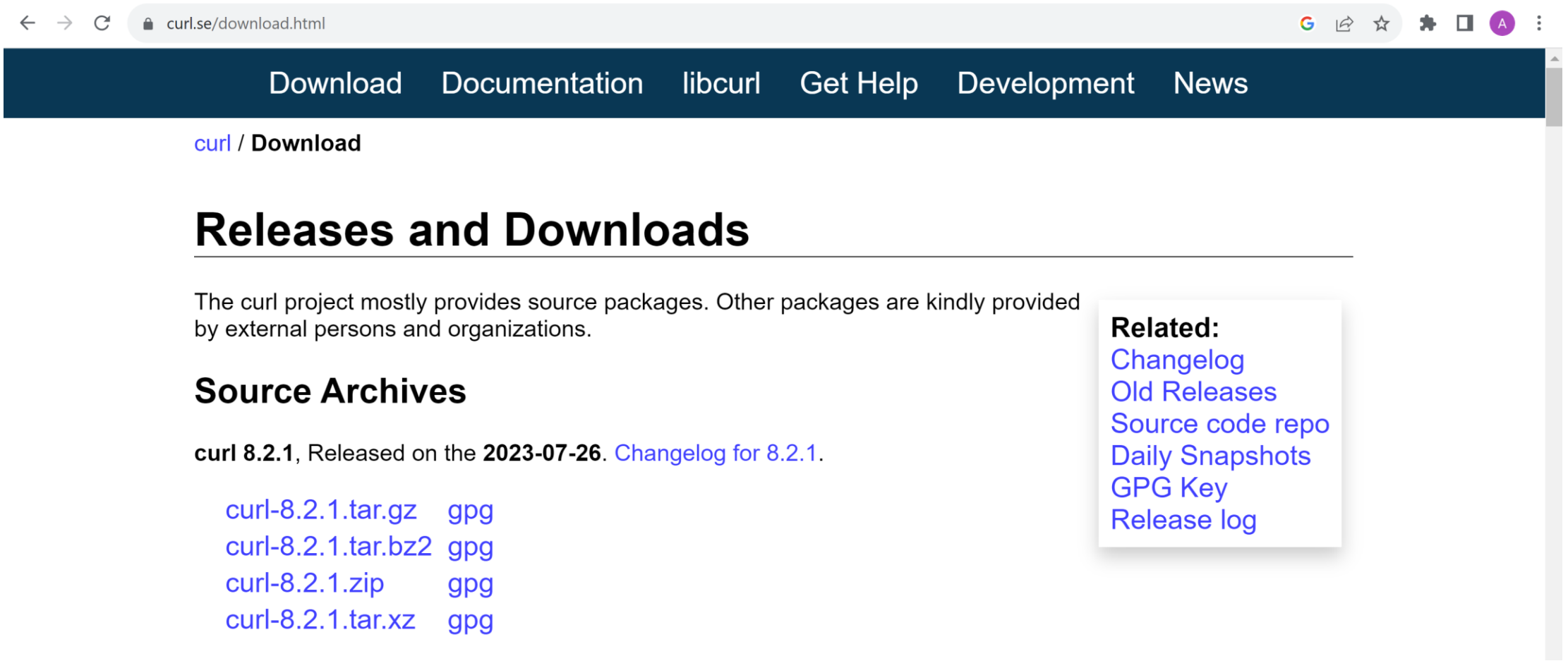
Test using Curl
Test using Postman
OpenApi – Swagger UI

$ curl … c:> ?? PS1> Invoke-WebRequest

Curl on Power-Shell = alias for Invoke-WebRequest
… different command line arguments / result formatting !



https://curl.se/download.html



$ curl sheet cheat

-v verbose

\>

\> Sent

<

< Received

--silent

-X GET,POST,PUT,DELETE

Default: GET when no -d
Or POST when –d ’request body’

-d ’request body’

@ : request body from file name
Example: $ curl –X POST http://loc @file.json

@- : request body from stdin
Example: $ cat << EOF
{ «field »: 123 }
EOF | curl –X POST http://loc @-

-H header:value

Example:
-H Content-Type:application/json
-H Accept:application/json

-u user:password

-x proxy or export HTTPS_PROXY

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

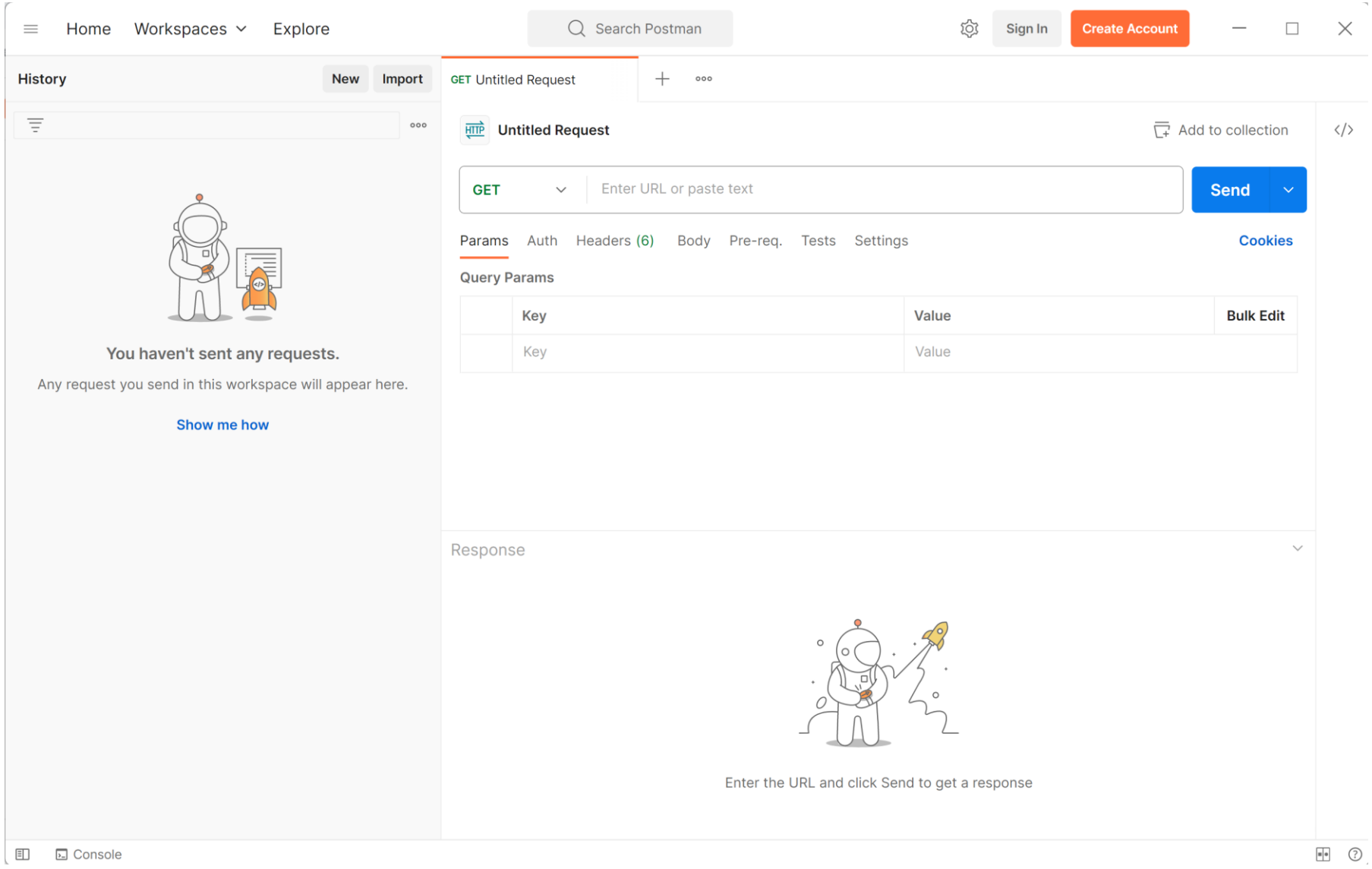
- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

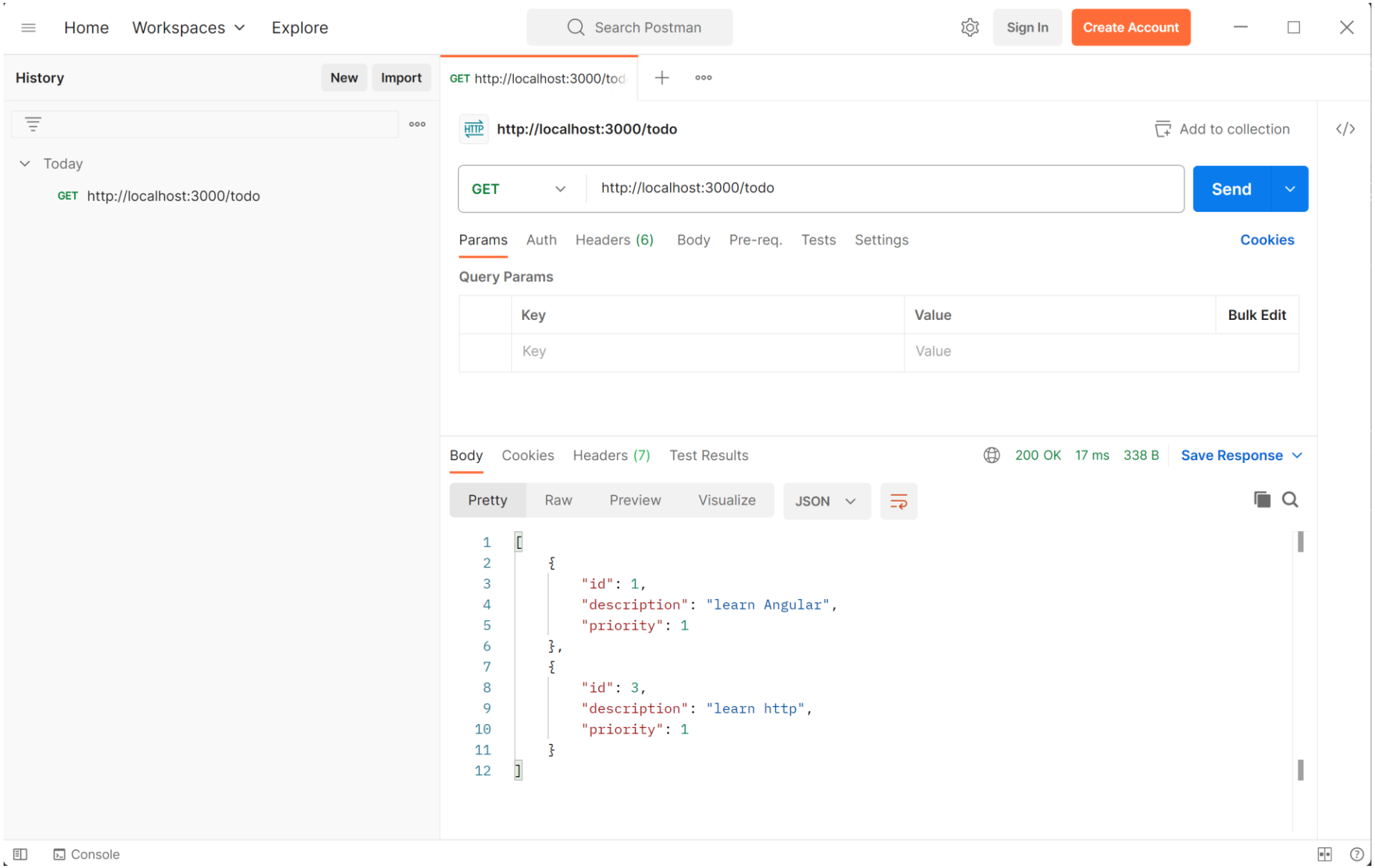
Test using Postman

OpenApi – Swagger UI

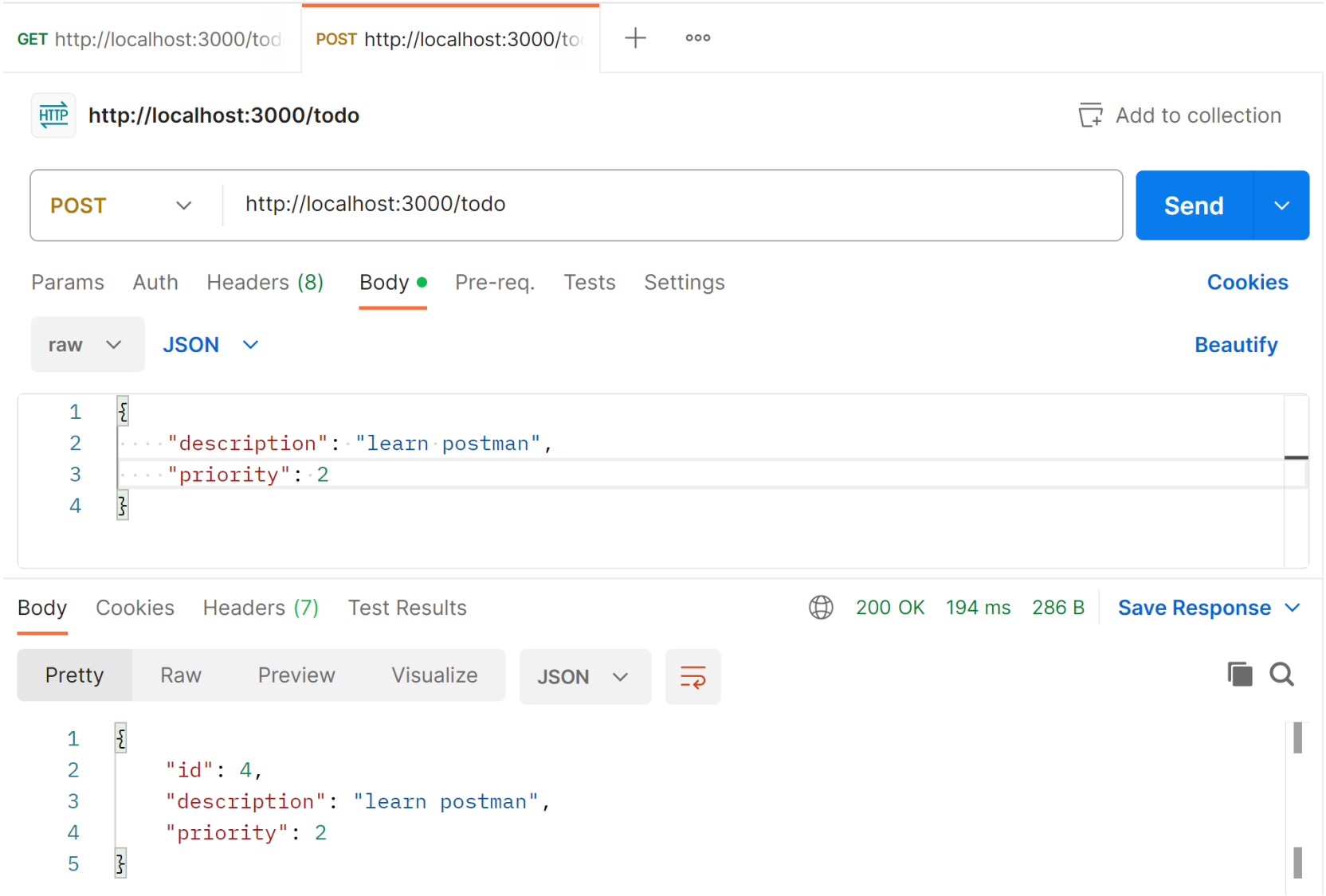
https://www.postman.com/downloads/



Postman http GET /todo



Postman http POST /todo req.body { json}



Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :
- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman

OpenApi – Swagger UI

Swagger (OpenAPI)

https://www.npmjs.com/package/swagger-ui-express

+

https://www.npmjs.com/package/swagger-express-router

```
$ npm i swagger-ui-express swagger-jsdoc
[                    ] / reify:z-schema: http fetch GET 200 https://registry.npmjs.org/z-schema/-/z-schem
```

Configuring express + swagger ..

```
3   const swaggerJsdoc = require("swagger-jsdoc");
4   const swaggerUi = require("swagger-ui-express");
```

```
155 const swaggerOptions = {
156   definition: {
157     openapi: "3.1.0",
158     info: { title: "Todo Rest API" },
159     servers: [ { url: "http://localhost:3000" } ]
160   },
161   apis: ["./index.js"],
162 };
163 const swaggerDoc = swaggerJsdoc(swaggerOptions);
164 console.log("swagger doc from annotated js:", swaggerDoc);
165
166 app.use('/api-docs', swaggerUi.serve);
167 app.get('/api-docs', swaggerUi.setup(swaggerDoc));
```

Adding Js Doc: /** @openapi .. */

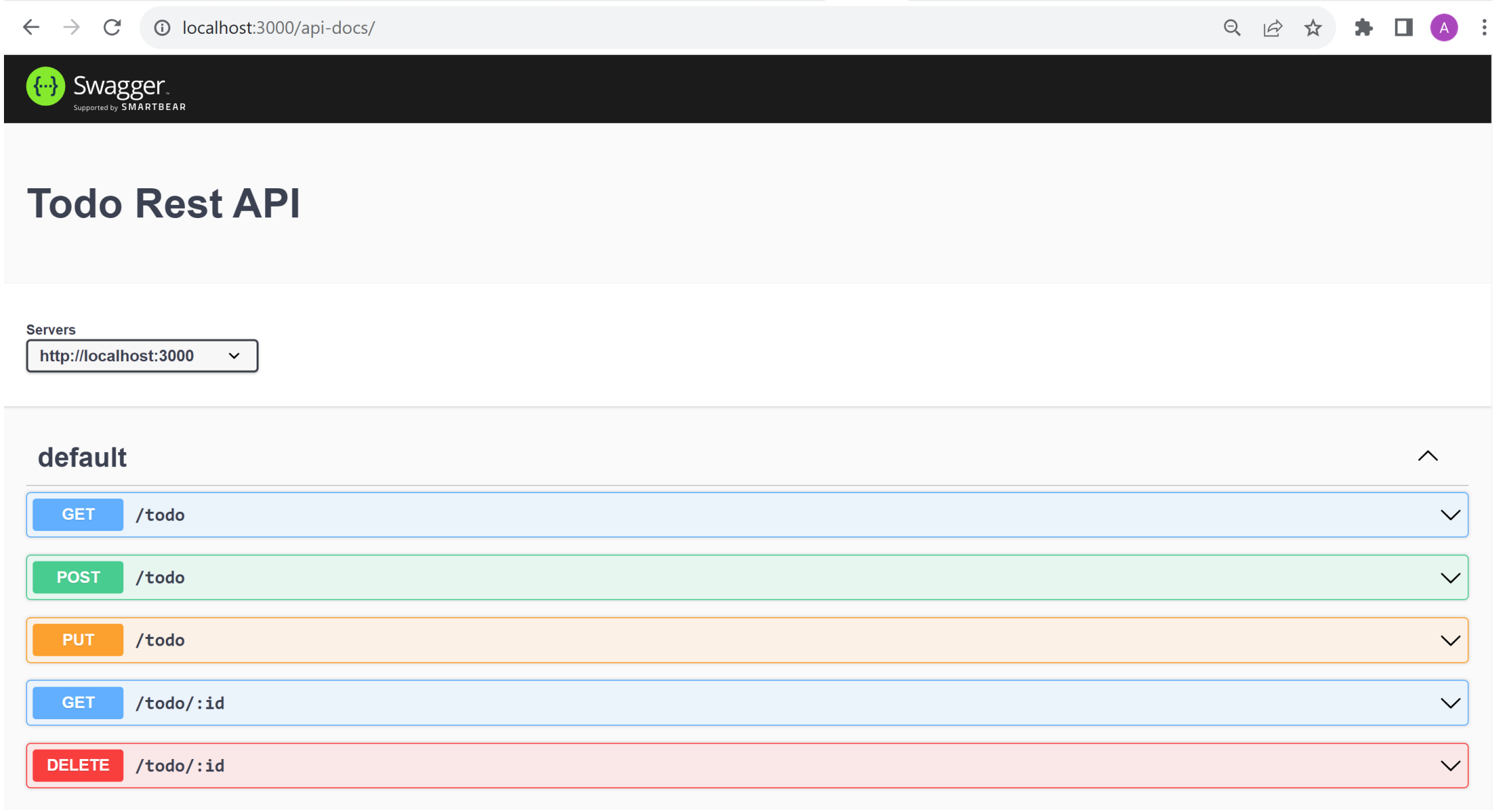
```
31  /**
32   * @openapi
33   * /todo:
34   *   get:
35   *     description: list of todos
36   *     responses:
37   *       200:
38   *         description: ok
39   */
40  app.get('/todo', function (req, res) {
41    res.send(todos)
42  });
```

Console.log swagger spec from js doc

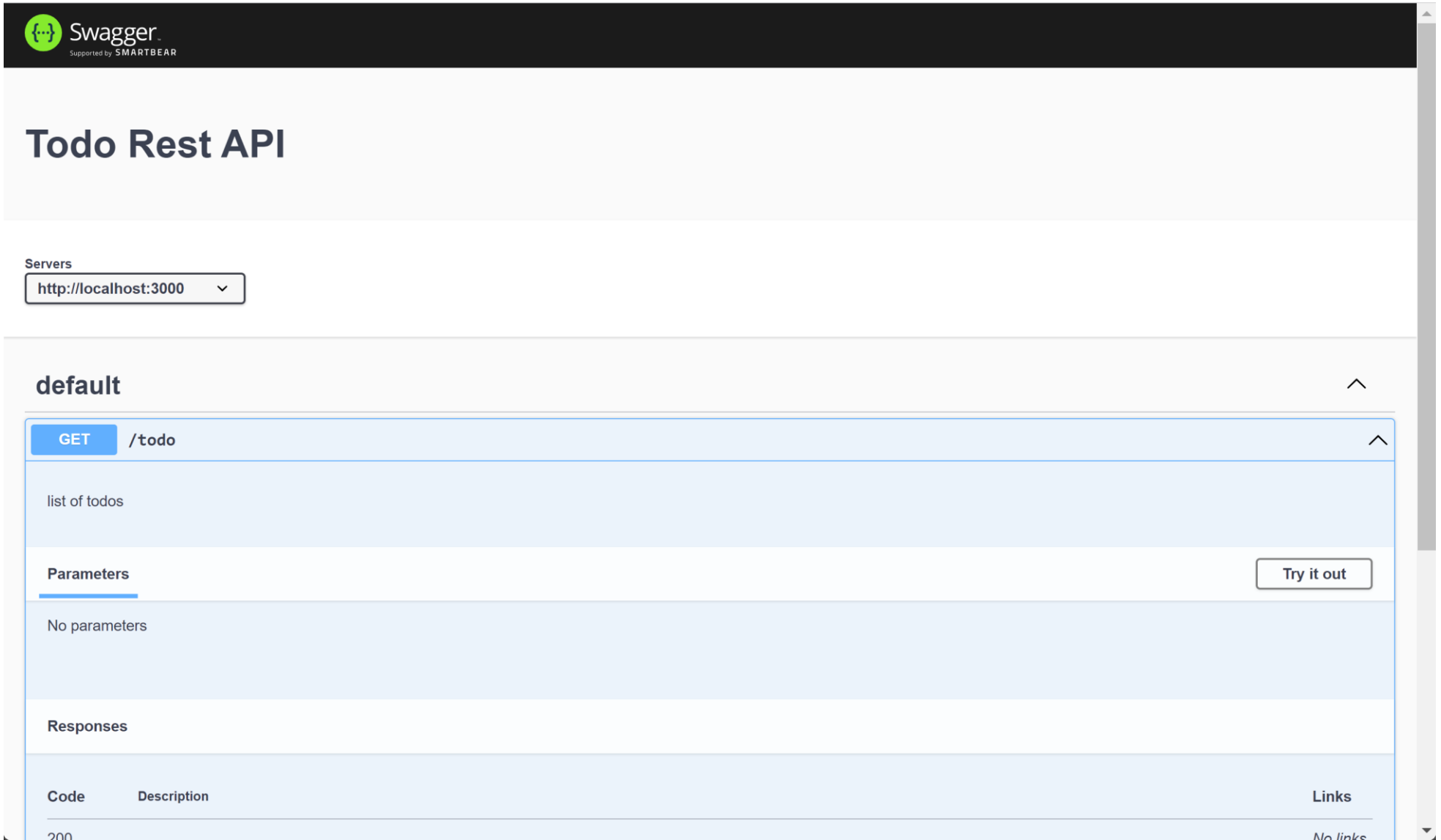
```
const swaggerDoc = swaggerJsdoc(swaggerOptions);
console.log("swagger doc from annotated js:", swaggerDoc);
```

```
$ node index.js
swagger doc from annotated js: {
  openapi: '3.1.0',
  info: { title: 'Todo Rest API' },
  servers: [ { url: 'http://localhost:3000' } ],
  paths: {
    '/todo': { get: [Object], post: [Object], put: [Object] },
    '/todo/:id': { get: [Object], delete: [Object] }
  },
  components: {},
  tags: []
}
```

Swagger-UI : http://localhost:3000/api-docs/



Test using Swagger-UI « Try it out » .. « Execute »



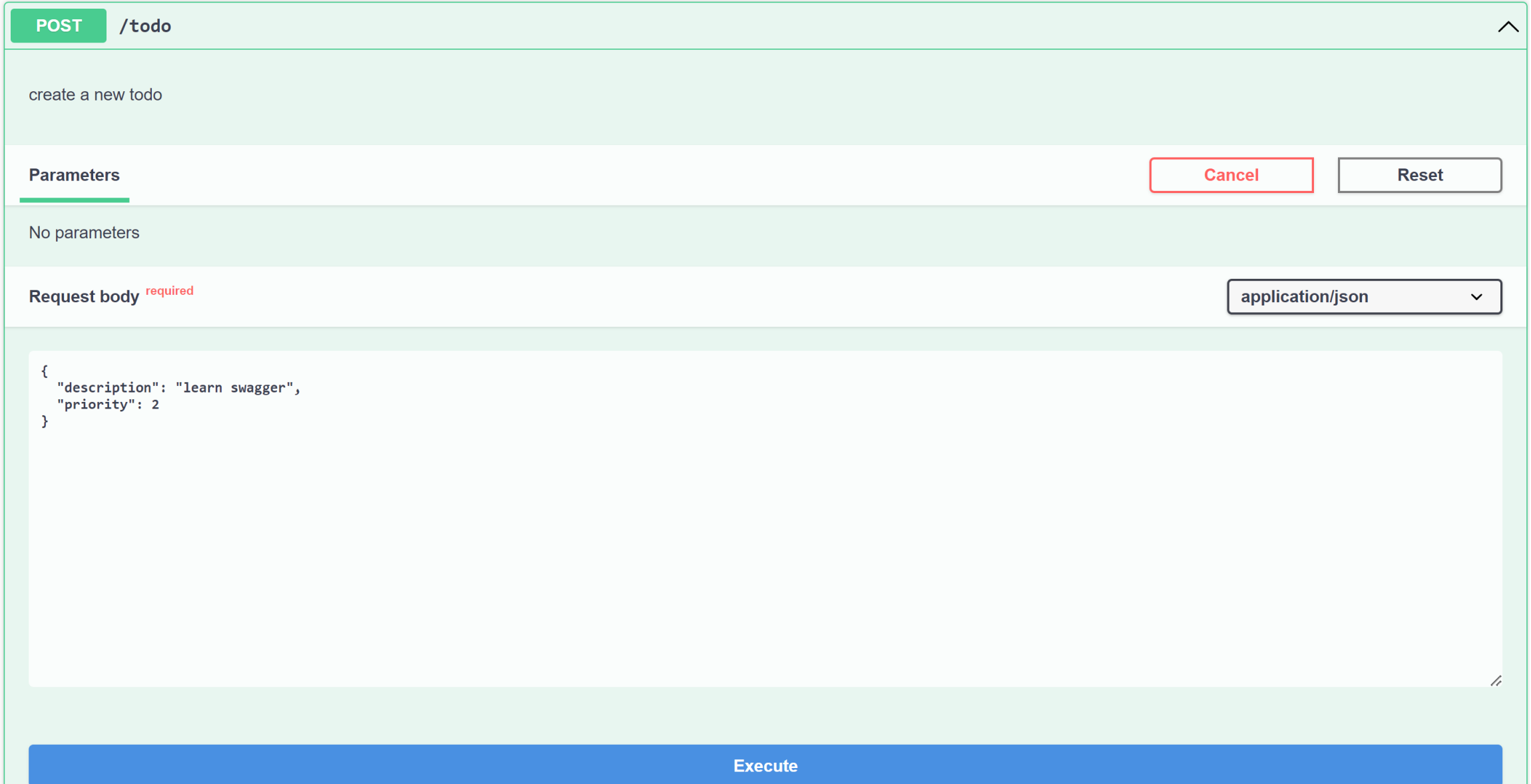
@openapi GET « /todo/{id} » parameters for express « /todo/:id »

```
46  /**
47   * @openapi
48   * /todo/{id}:
49   *   get:
50   *     description: todo by id
51   *     parameters:
52   *      - name: id
53   *        in: path
54   *        description: id of the todo entity
55   *        required: true
56   *        type: integer
57   *     produces:
58   *       - application/json
59   *     responses:
60   *       200:
61   *         description: ok
62   *       404:
63   *         description: Todo entity not found
64   */
65  app.get('/todo/:id', function (req, res) {
```

@openapi POST requestBody .. content .. schema

```
/**
 * @openapi
 * /todo:
 *   post:
 *     description: create a new todo
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             type: object
 *             properties:
 *               description:
 *                 type: string
 *                 description: description of the todo entity
 *               priority:
 *                 type: number
 *                 description: priority of the todo entity
 *     produces:
 *       - application/json
 *     responses:
 *       200:
 *         description: ok
 */
app.post('/todo', function (req, res) {
```

Test Swagger POST with requestBody .. Required json



Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl
Test using Postman
OpenApi – Swagger UI

Next Steps :
Create Angular FrontEnd App

Take Away

Simple to code a http server
.. even in NodeJS
(better in Java – Springboot
    IDE / Debugger / Type checking / Libraries / ..
    more valuable for work in companies
)

Rest endpoint = http server using url + json
                + conventions for state transfer on GET,POST,PUT,DELETE

Use Curl / Postman / OpenApi Swagger

Questions

Next Steps …