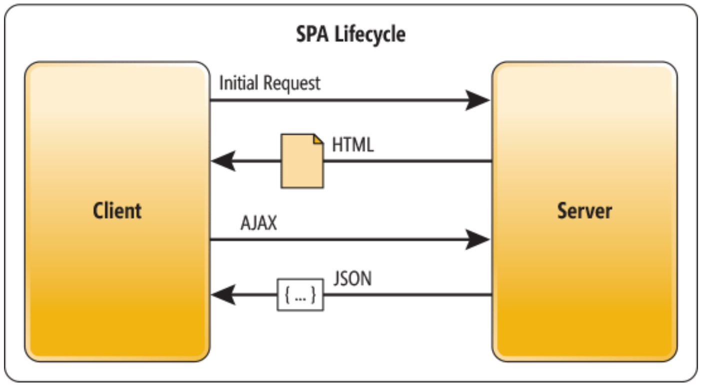
CS 312 Software Development

Servers

Obtaining data for our application

reminder



Wasson, Microsoft

A simple HTTP server

Manually construct the response

```
const http = require('http’);
const server = http.createServer((request, response) => {
  response.writeHead(200, { 'Content-Type': 'text/plain' });
  response.end(“Don’t Panic”);
}).listen(5042);
console.log( 'Listening on port %d', server.address().port );
```

In action:

```
$ curl http://localhost:5042/
Don’t Panic
```

API

routes and endpoints

Endpoints

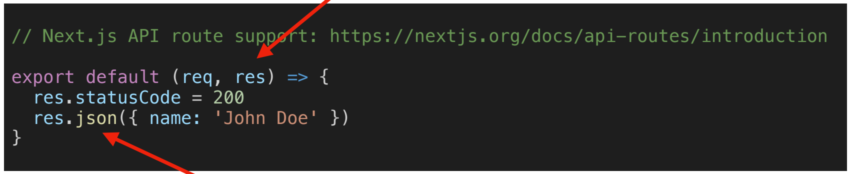
Route	HTTP Verb	Controller Action
/api/films	POST	Create new film from request data
/api/films/:id	GET	Read data of film with id == :id
/api/films/:id	PUT	Update film with id == :id from request data
/api/films/:id	DELETE	Delete film with id == :id
/api/films	GET	List (read) all films

```
const http = require('http’);
const server = http.createServer((request, response) => {
  const path = url.parse(request.url, true).query;
  if (path === ‘/api/films’ && request.method === ‘GET’){
   …
  }
}).listen(5042);
console.log( 'Listening on port %d', server.address().port );
```

Next.js API routes

pages/api/hello.js

Function which accepts the request and response objects



```
// Next.js API route support: https://nextjs.org/docs/api-routes/introduction

export default (req, res) => {
  res.statusCode = 200
  res.json({ name: 'John Doe' })
}
```

convenience function for returning JSON

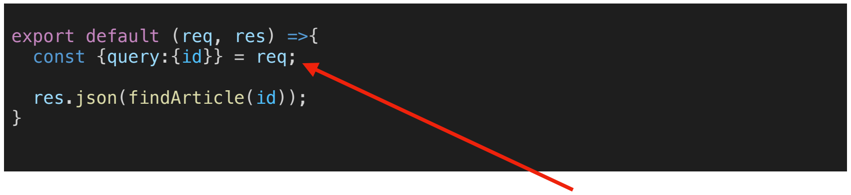
Available at: localhost:3000/api/hello

Next.js API routes

dynamic routes

pages/api/article/[id].js

square brackets indicates a dynamic route



```
export default (req, res) =>{
  const {query:{id}} = req;

  res.json(findArticle(id));
}
```

req.query contains the portion of the URL that maps to the id

Available at: localhost:3000/api/article/2

Next.js API routes

file structure

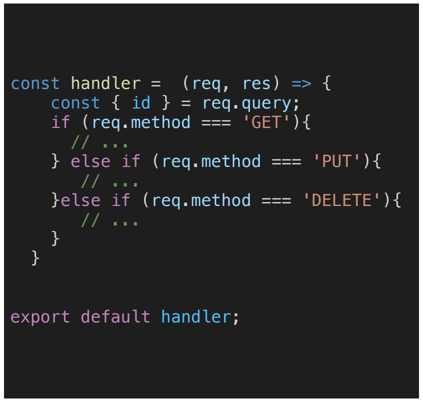
Route	File	Controller Action
/api/films	/pages/api/films/index.js	Create new film from request data
/api/films/:id	/pages/api/films/[id].js	Read data of film with id == :id
/api/films/:id	/pages/api/films/[id].js	Update film with id == :id from request data
/api/films/:id	/pages/api/films/[id].js	Delete film with id == :id
/api/films	/pages/api/films/index.js	List (read) all films

OR

Route	File	Controller Action
/api/films	/pages/api/films.js	Create new film from request data
/api/films/:id	/pages/api/films/[id].js	Read data of film with id == :id
/api/films/:id	/pages/api/films/[id].js	Update film with id == :id from request data
/api/films/:id	/pages/api/films/[id].js	Delete film with id == :id
/api/films	/pages/api/films.js	List (read) all films

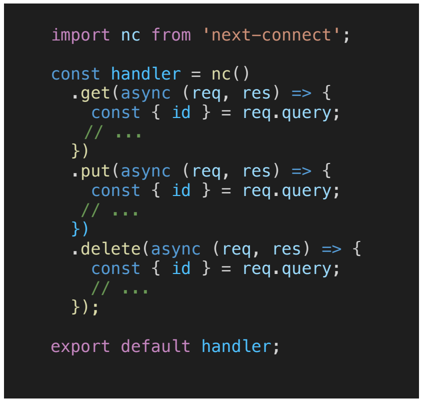
next-connect

Default API routes



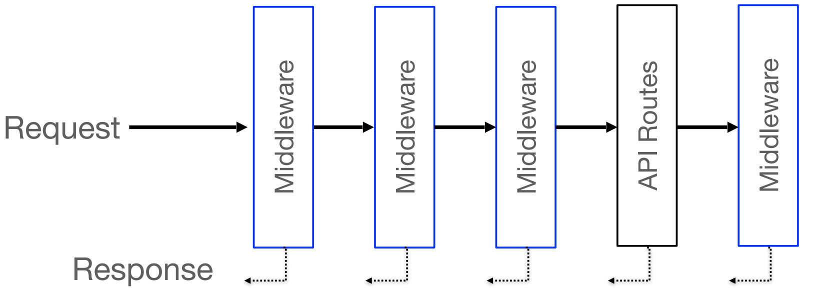
```
const handler =  (req, res) => {
    const { id } = req.query;
    if (req.method === 'GET'){
      // ...
    } else if (req.method === 'PUT'){
       // ...
    }else if (req.method === 'DELETE'){
       // ...
    }
  }


export default handler;
```

Using next-connect



```
import nc from 'next-connect';

const handler = nc()
  .get(async (req, res) => {
    const { id } = req.query;
   // ...
  })
  .put(async (req, res) => {
    const { id } = req.query;
   // ...
  })
  .delete(async (req, res) => {
    const { id } = req.query;
    // ...
  });

export default handler;
```

Middleware



Middleware modify or respond to request

Built in middleware

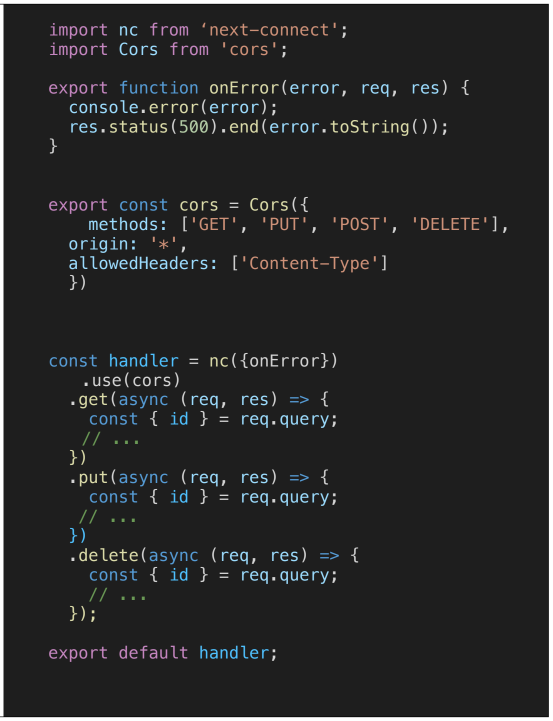
- req.query - builds the query and leaves it in req.query
- req.body - parses the body of the request for us in the format specified in the headers
- req.cookies - contains the cookies sent with the request

Aspect-oriented Programming (AOP)

- Design pattern for implementing “cross-cutting” concerns
  - Middleware is an example of AOP
- “Cross cutting” concerns are those that affect many parts (or concerns) of the code
  - Many requests require body parsing
- AOP is a general set of techniques for DRYing up “cross cutting” concerns

Middleware

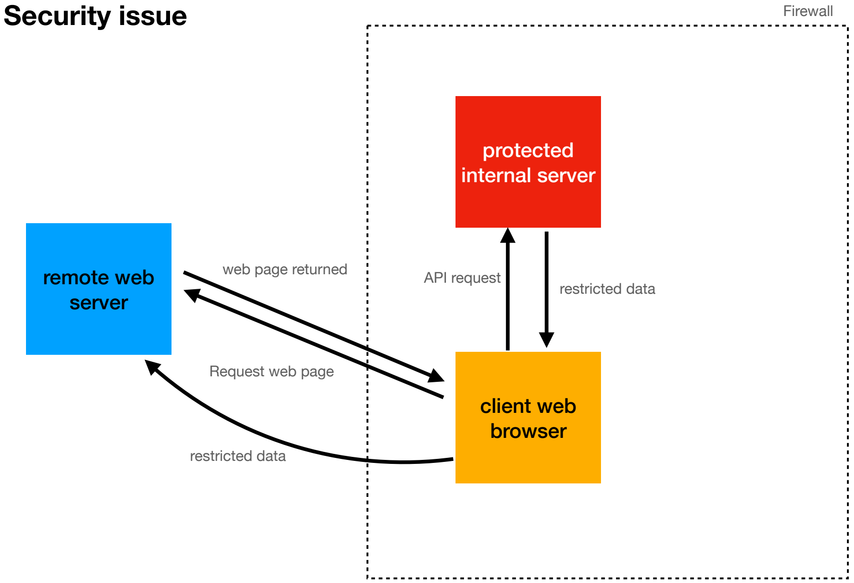
example



```
import nc from ‘next-connect';
import Cors from 'cors';

export function onError(error, req, res) {
  console.error(error);
  res.status(500).end(error.toString());
}


export const cors = Cors({
    methods: ['GET', 'PUT', 'POST', 'DELETE'],
  origin: '*',
  allowedHeaders: ['Content-Type']
  })



const handler = nc({onError})
   .use(cors)
  .get(async (req, res) => {
    const { id } = req.query;
   // ...
  })
  .put(async (req, res) => {
    const { id } = req.query;
   // ...
  })
  .delete(async (req, res) => {
    const { id } = req.query;
    // ...
  });

export default handler;
```

CORS

Security issue


Firewall
protected internal server
remote web server
web page returned
Request web page
API request
restricted data
client web browser
restricted data