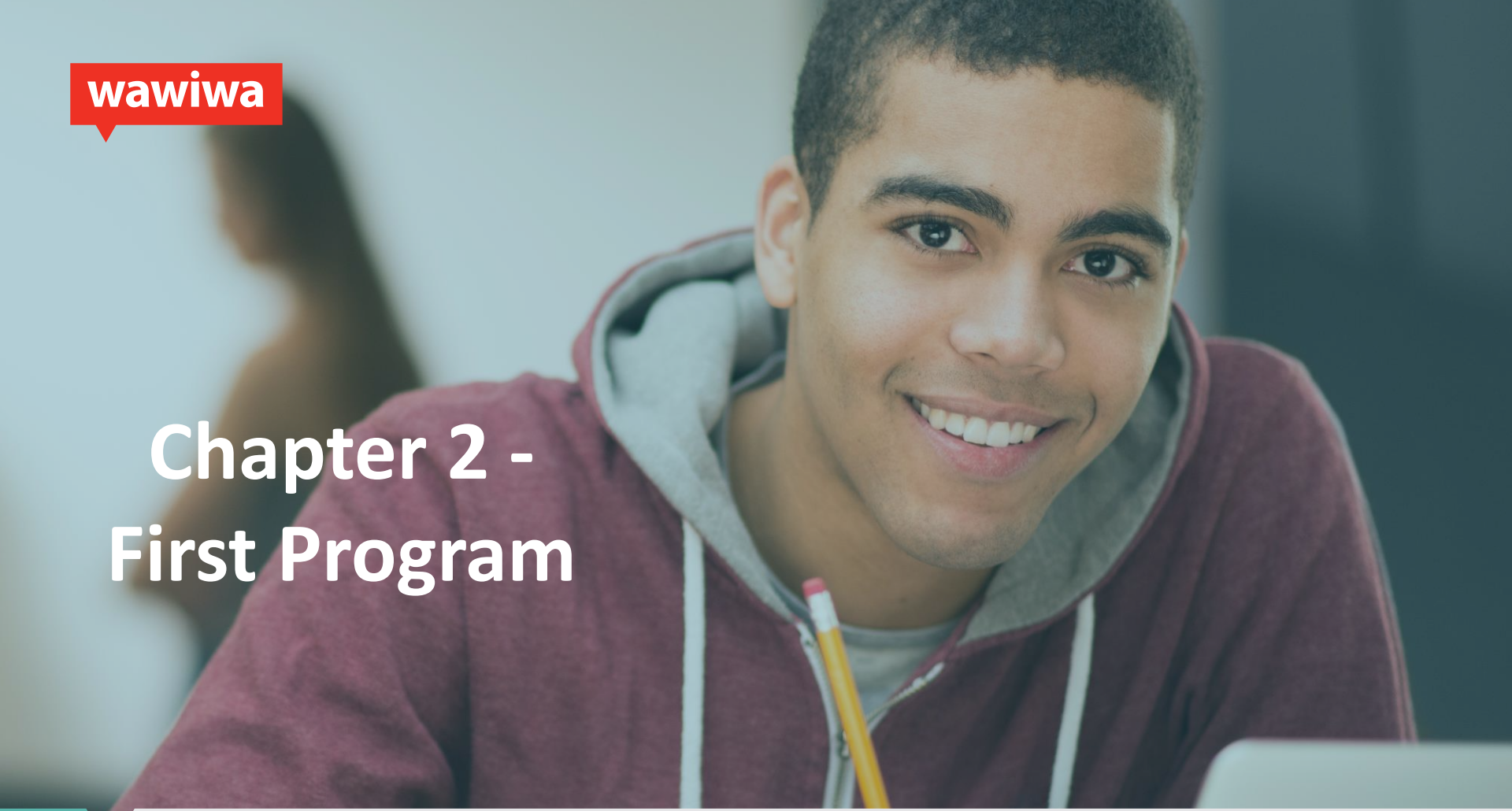
wawiwa

NODEJS Practice Booklet

wawiwa



Chapter 2 - First Program

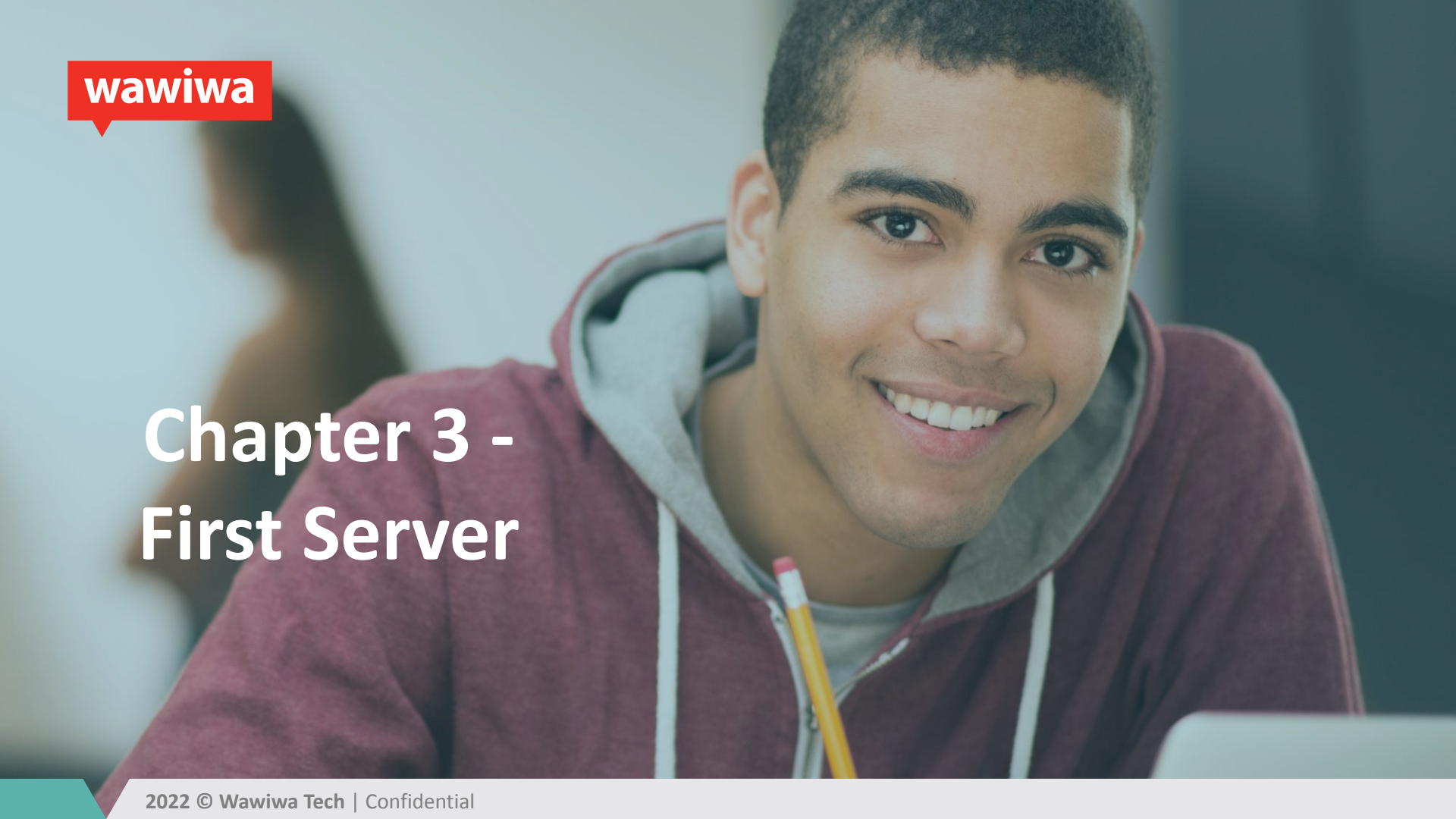
Do it yourself 1

Export a Name Module. It has 2 functions:

1. A function that receives first and last name and returns it with “hello”.
2. A function that receives sender name , message and subject and prints them.

For Example: “Jack”, “Hello, how are you?”, “Dan”, the return message is: “Dan, you got a new message from Jack: Hello, how are you?”

Create a program and use it.



Chapter 3 - First Server

2022 © Wawiwa Tech | Confidential

wawiwa

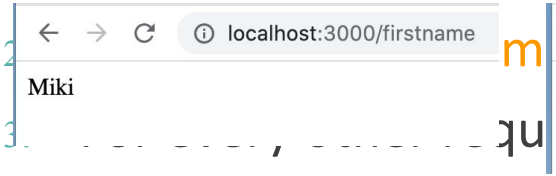
Do it yourself 1

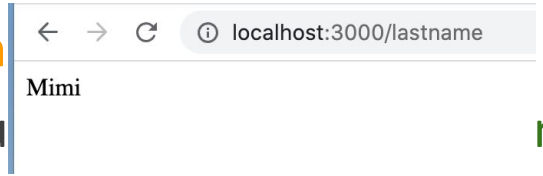
Create a server that responds to 127.0.0.1:3000 with your first name and last name

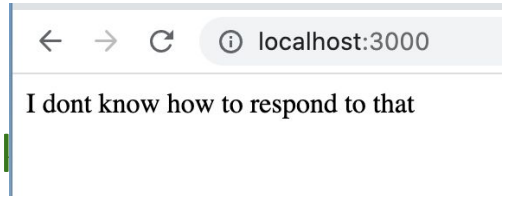
Do it yourself 2

Build a server on port 3000 that responds to the user in the following way:

1. For the url '/firstname' - return "Miki"
2. For the url '/lastname' - return "Mimi"
3. For any other request - return "I don't know how to respond to that"





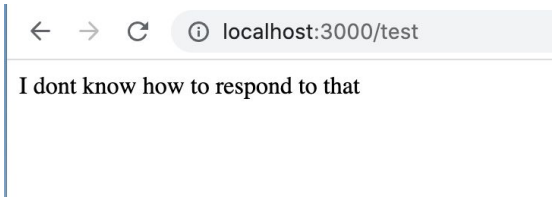


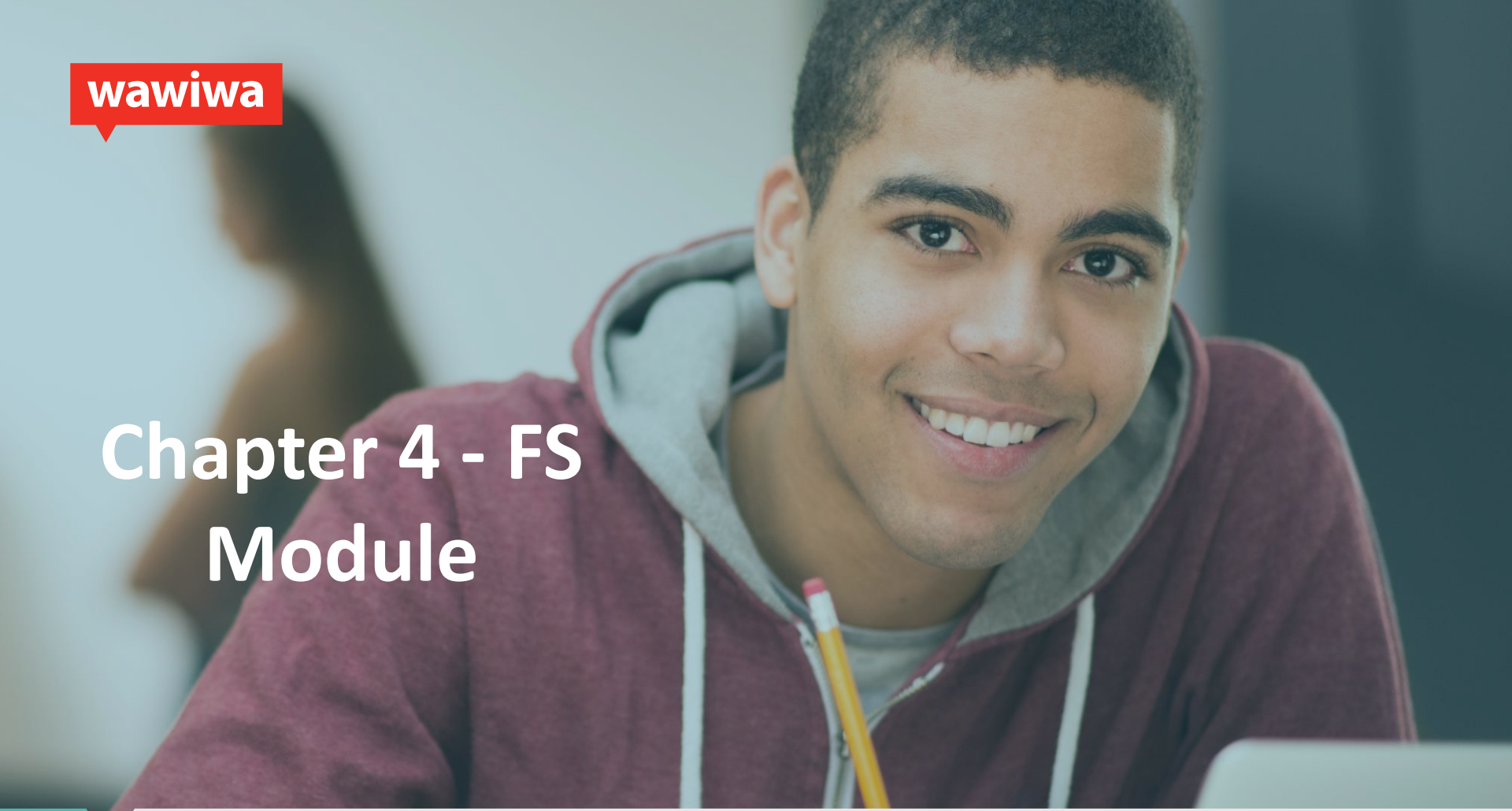
wawiwa

Do it yourself 3

Build a server on port 3000 that responds to the user in the following way:

1. For the url '/params' - return all parameters
2. For every other request responds with "I don't know how to respond to that"





Chapter 4 - FS Module

wawiwa

Do it yourself 1

Write a program that reads **synchronous** from a file named test.txt and prints the text to the console

wawiwa

Do it yourself 2

Write a program that reads **asynchronous** from a file called test.txt and prints the text to the console

wawiwa

Do it yourself 3

Create a program in node that writes **synchronously** to a file named test.txt with your full name.

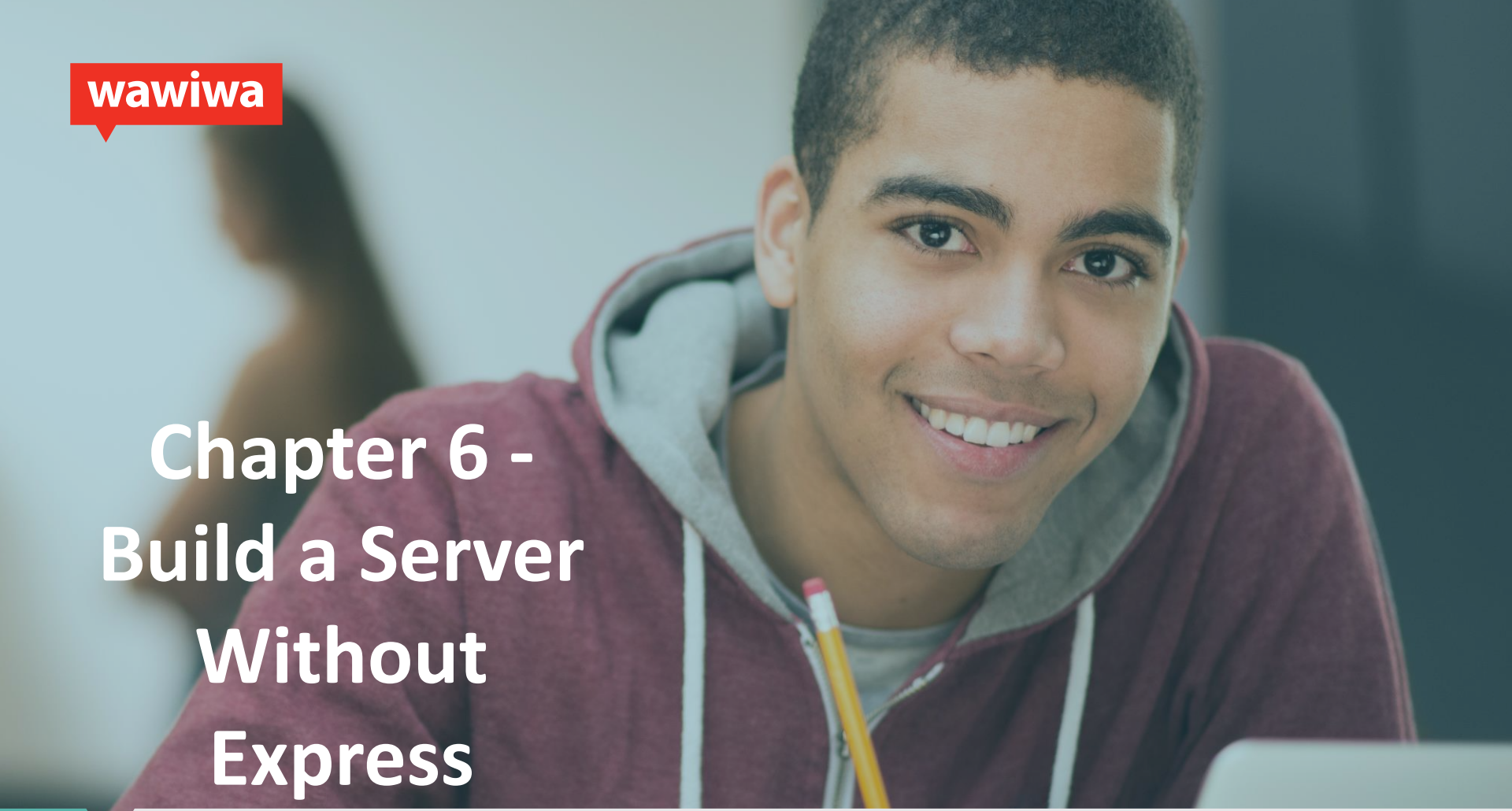
wawiwa

Do it yourself 4

Create a program in node that writes **asynchronously** to a file named test.txt with your full name.

wawiwa

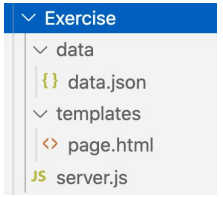
Chapter 6 - Build a Server Without Express



wawiwa

Do it yourself 1

Add this file to data folder



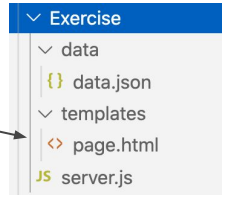
1. Create a folder with 2 directories
    a. data that contain data.json

```
[
    {
      "product": "smile",
      "description": "Yellow Smile",
      "price": "500",
      "image": "😄"
    }
 ]
```

b. templates - contain page.html - next slides content of page.html

wawiwa

Add this file to templates folder



page.html

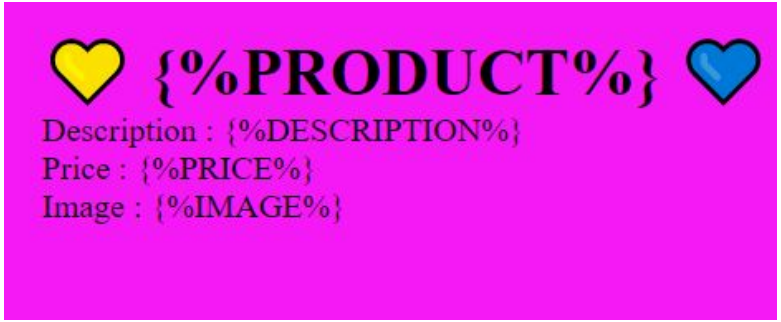
```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device
   <meta http-equiv="X-UA-Compatible" content=
   <title>Products</title>
   <style>
     *
     {
       margin: 0;
       padding: 0;
       box-sizing: inherit;
     }

     html {
       box-sizing: border-box;
     }
```

```
     body {
       background:hsl(300, 92%, 53%);
     }
     .container{
       padding:20px;
     }
     .cards{ial-scale=1.0"  />
       padding: 5px;
     }
     .card{
       border: 5px solid royalblue;
       width: 25%;
       border-radius: 10px;
       padding: 5px;
       font-size: 20px;
       margin:10px 10px;
       float: left;
     }
   </style>
 </head>
```

page.html continue

You can use any sign. (It is not mandatory to use the% sign)

```
<body>
  <div class="container">
    <h1>💛 {%PRODUCT%} 💙</h1>
    <div>
      <p>Description : {%DESCRIPTION%} </p>
      <p>Price : {%PRICE%} </p>
      <p>Image : {%IMAGE%} </p>
    </div>
  </div>
</body>
</html>
```



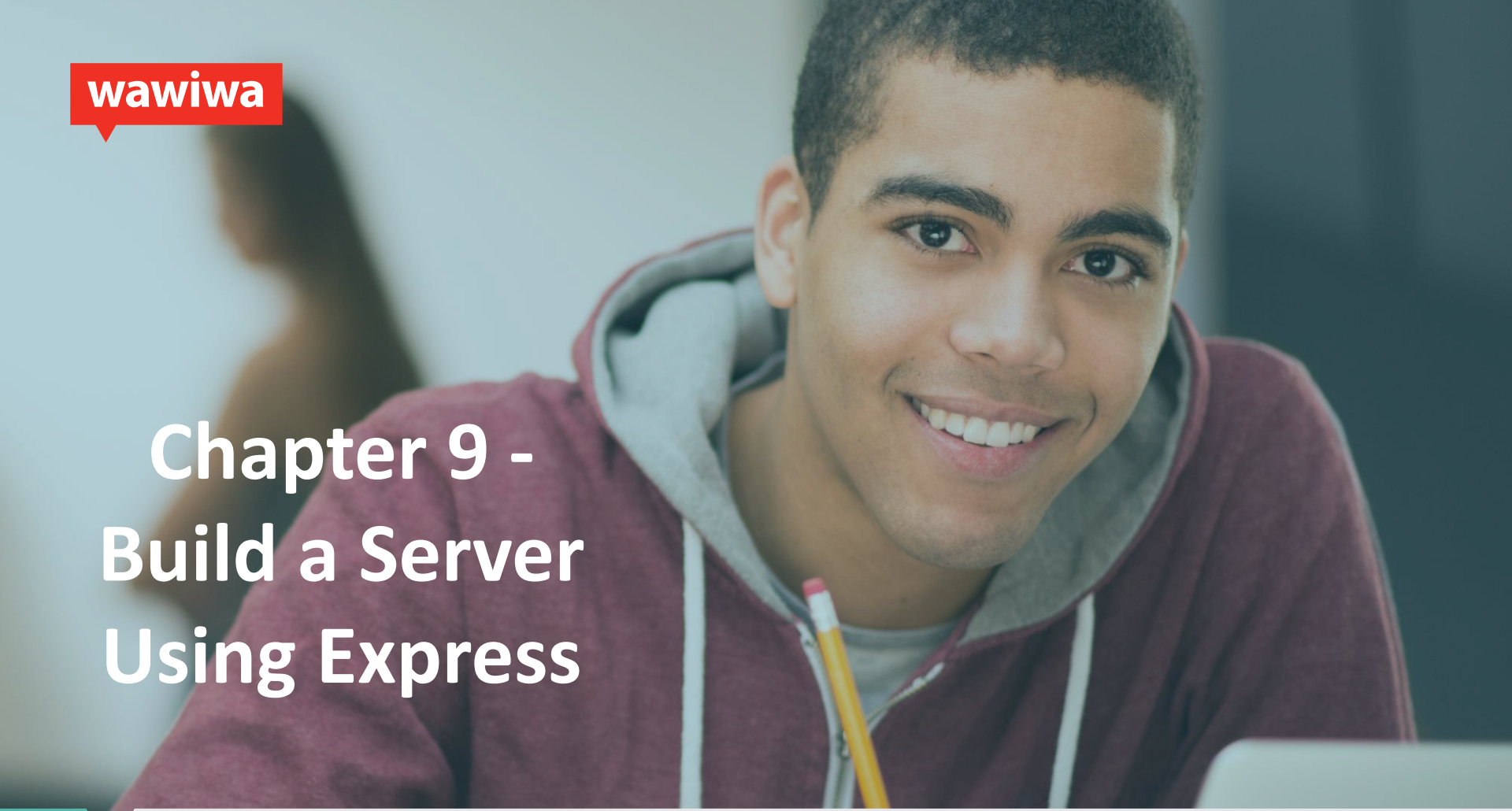
Exercise continue

Create a server that listens to port 3000 and responds with the relevant data.json.

response

wawiwa

Chapter 9 - Build a Server Using Express



Do it yourself 1 - Build A Server Using Express

Create a server that responds to the following requests:

Request	Response	Method
localhost:3000/api/food	Your favorite food	get
localhost:3000/random/number	Random Number between 0 to 10	get
localhost:3000/courses/n1ton2	Random Number between 100 to 2000	Post

wawiwa

do it yourself 2 - data/dancers.json

```
[
    {
        "id":0,
        "firstname":"Ashley",
        "lastname":"Pitt",
        "age":16,
        "specialty":"ballet"
    },
    {
         "id":1,
         "firstname":"Jon",
         "lastname":"Anthony",
```

wawiwa

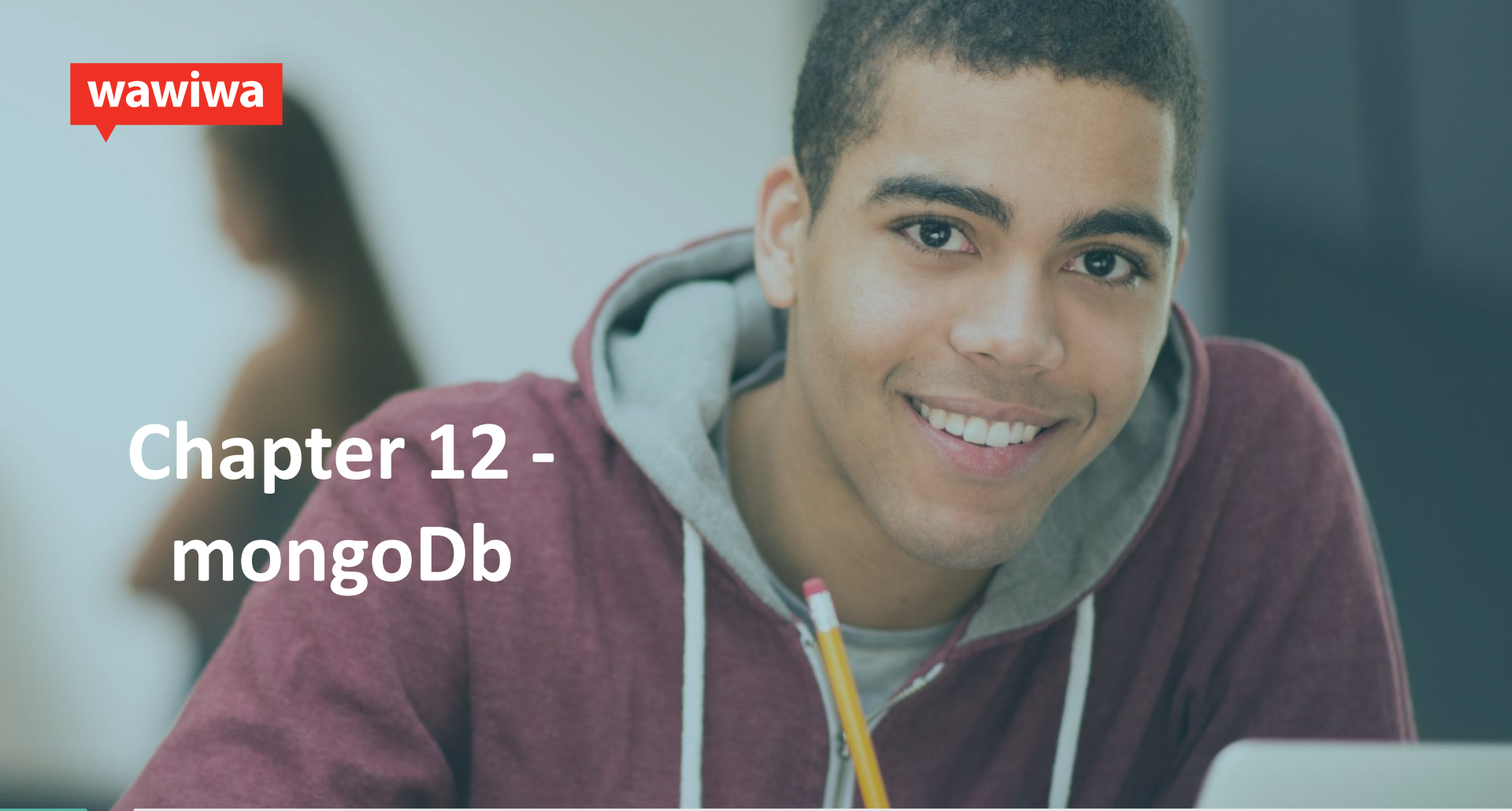
Do it yourself 2 - Build A Server Using Express

Create an express server that responds for the following requests:

Request	Response	Method	Comment
`/api/v1/dancers`	All dancers	get	Get all dancers from dancers.json
`/api/v1/dancers`	The created dancer	Post	Create new dancer
`/api/v1/dancers/:id`	Dancer by id	Get	Get dancer by id
`/api/v1/dancers/:id`	Update dancer	Patch	Update dancer by id
`/api/v1/dancers/:id`	delete dancer	Delete	Delete dancer by id

wawiwa

Chapter 12 - mongoDb



Do it yourself 1

The sequence available in the next slides (till the end of the chater) will only work if we perform all the steps described in the previous slides - connection to DB,try, Catch etc

1. Open an account in mongodb
2. After registering, make sure you have a user in the account's user list who has access to the account
3. Make sure your ip is in the ip list if not then add it
4. Create a js file named red1.js
5. Click connect and then connect your application and select nodejs version 4 Copy the code and paste it in red1.js Run the file by node red1

wawiwa

Do it yourself 2

Create a product object with the following attributes in the red1.js file:

```
let personDocument = {
        "name": { "first": "Lisa", "last": "Mine" },
        "birth": new Date(1971, 8, 22),
        "death": new Date(1967, 9, 9),
        "contribs": [ "Turing machine"],
        "views": 145000
    }
```

Add the correct commands to put this person into mongodb using the insert command
Run the red1 file and make sure a person logs in

wawiwa

Do it yourself 3

Open a file named red3.js and write a suitable code that will print to the console all the records that are in mongodb that has 135000 views.
Help with the command:

```
find().toArray();
```

wawiwa

Do it yourself 4

Open a file named red4.js and write a suitable code that will print to the console all the records that are in mongodb that has between 50000 and 100000 views.
Help with the command:

```
find().toArray();
```

wawiwa

Do it yourself 5

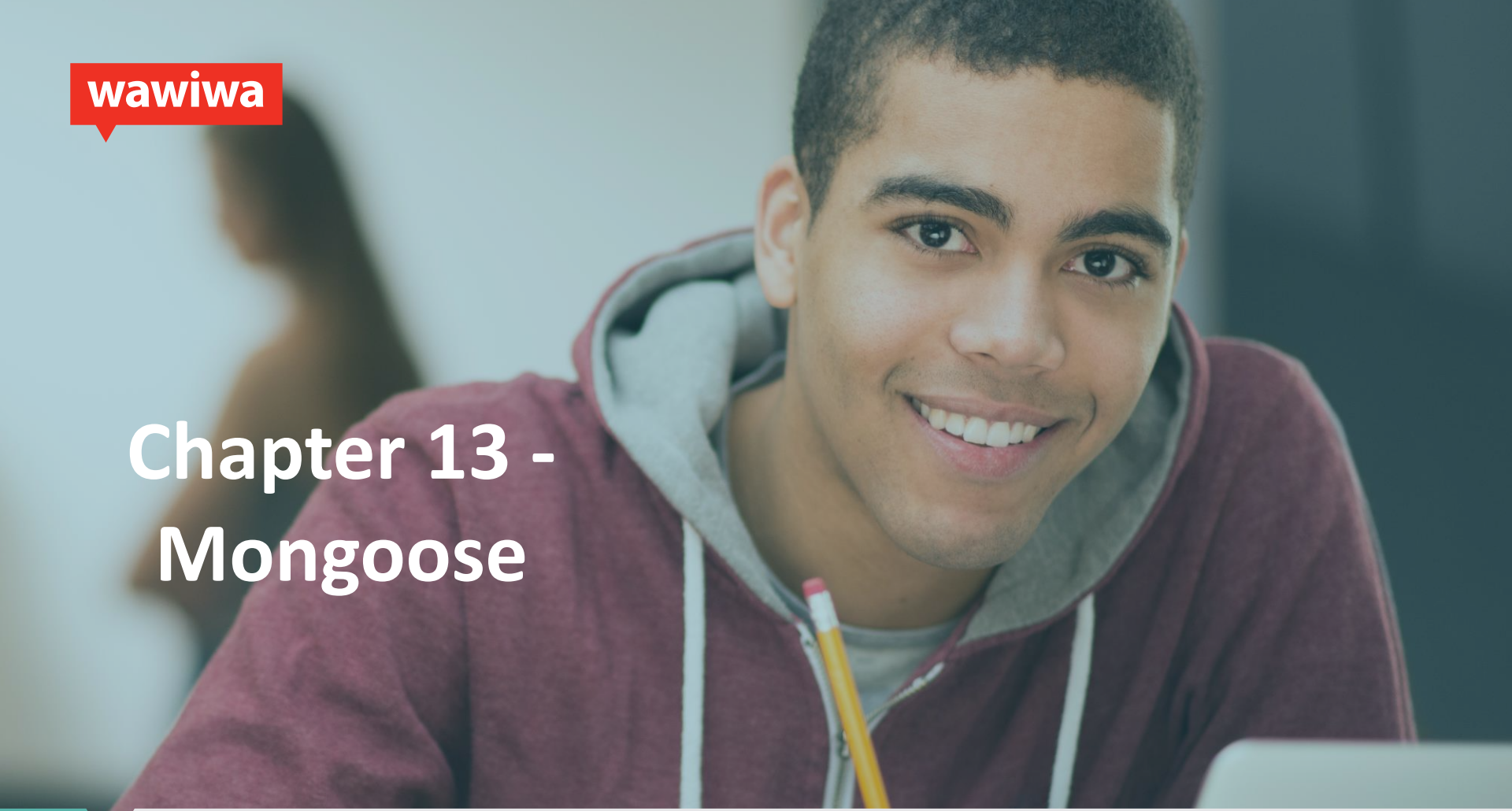
Open a file named red5.js and write a suitable code that will print to the console all the records, limit the number of records to 4.
Help with the command:

```
find().toArray();
```

wawiwa

Do it yourself 6

Open a file named red6.js and write an appropriate code that will update the record that has 12000 views to 325011 Help with updateOne:



Chapter 13 - Mongoose

wawiwa

Do it yourself 1

1. Create a folder that contains an app.js file
2. Add to the folder a model named CityModel.js that contains the following fields: name, country, population
3. Add a new function that insert city to db
4. Run the application and insert 3 cities to db
5. Check in mongodb that the entries have been entered

wawiwa

Do it yourself 2

1. Add a new function to app.js to retrieve the data from the db
2. Rerun the application and navigate to localhost:3000/api/v1/cities

wawiwa

Do it yourself 3

1. Add a new function to app.js to retrieve the city by _id from the db
2. Rerun the application and navigate to localhost:3000/api/v1/cities/_ _ _ _ _ _ _ _ _ _ (select an id)

wawiwa

Do it yourself 4

1. Add a new function to app.js to delete city by _id
2. Rerun the application and navigate to localhost:3000/api/v1/cities/_ _ _ _ _ _ _ _ _ _ (select an id and run from postman delete request and delete the selected person)

wawiwa

Do it yourself 5

1. Update the function from exercise 2 and to filter data from db.
2. Rerun the application and navigate to localhost:3000/api/v1/persons/ and retrieve all cities with population 1000
3. Rerun the application and navigate to localhost:3000/api/v1/persons/ and retrieve all cities in country Italy

wawiwa

Do it yourself 6

1. Update the function from exercise 5 and to advance filter data from db.
2. Rerun the application and navigate to localhost:3000/api/v1/cities/ and retrieve all cities with population above 1000
3. Rerun the application and navigate to localhost:3000/api/v1/cities/ and add parameters to the query that retrieve all cities with population below 1000
4. Rerun the application and navigate to localhost:3000/api/v1/cities/ and add parameters to the query that retrieve all cities with population between 1000 to 2000