

RUMAD

Rutgers Mobile App Development

Overview

For this section, we'll be covering:

- Asynchronous Calls
- Requests
- Express endpoints

Review / Questions

- npm demo
- package / dependencies explanation



Review

NPM - How do we use it?

- The basic commands are:
 - **npm -v** to confirm the installation
 - **npm init** to initialize a folder as a local package
 - **npm install <package_name>** to install packages
 - `npm i <package_name>`

NPM - Configuration file

What are dependencies?

- “a piece of code—a library, a module, or a package—that a project requires to function correctly”

What is Express? (RECAP)

“Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.” ([expressjs](#))

Express provides us with functionality to:

- Route traffic
- Write and use middleware
- Implement API endpoints

The background is a solid dark red color. On the left side, there is a faint, stylized illustration of a smartphone. The phone's screen displays a white icon of a mask with two triangular eye cutouts. Below the screen, there are several white dots and a white lightbulb icon, suggesting a thought or idea. The word "Requests" is centered in the middle of the image in a large, white, bold, sans-serif font.

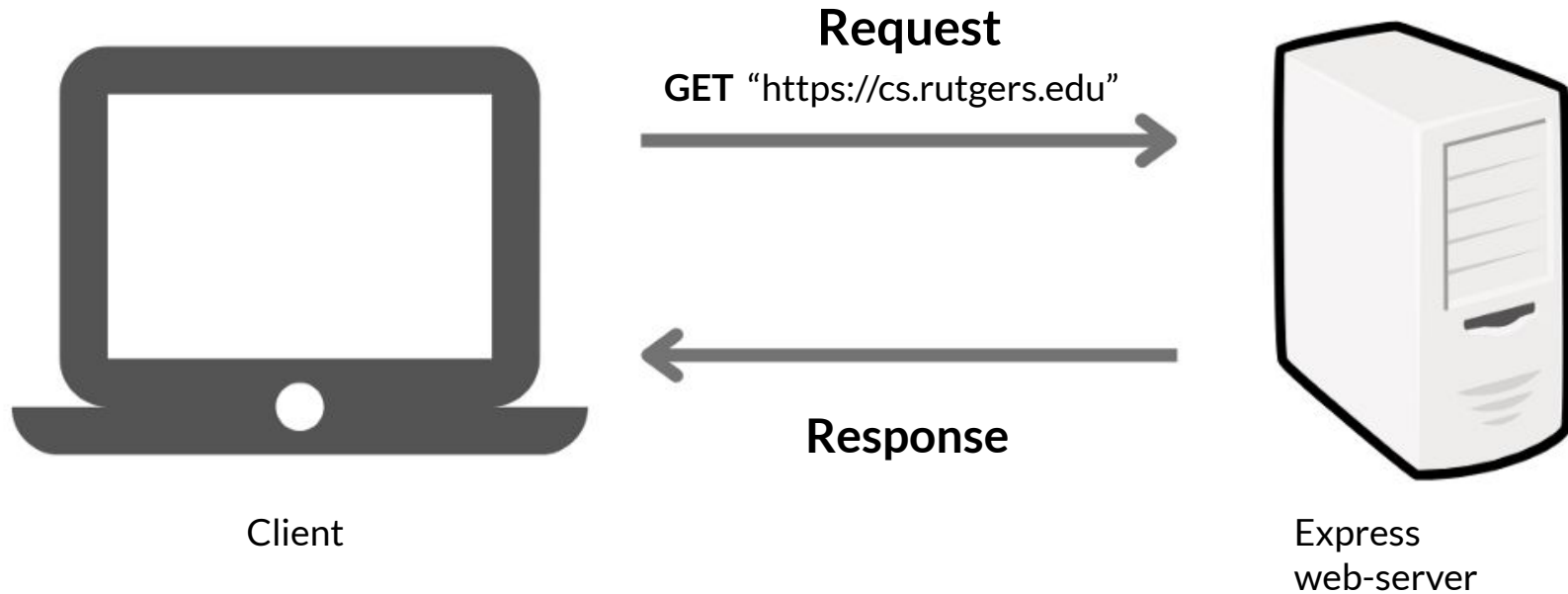
Requests

About web-applications

- A computer that stores and delivers the content for a website / API
- Common client is a web browser program
- Operates based on different requests, primarily:
 - GET
 - POST

There are more types, but I won't cover them. Feel free to ask me after or look up them:

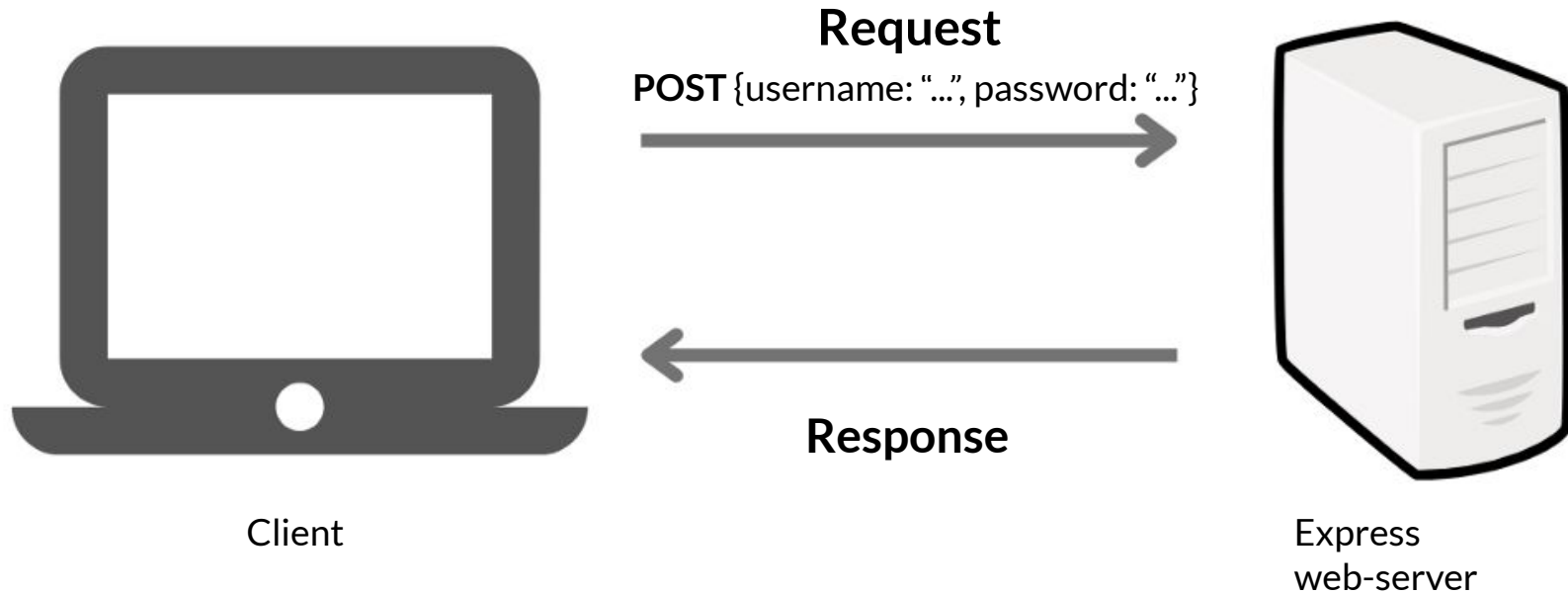
- HEAD
- PUT
- DELETE



Live example: <https://www.cs.rutgers.edu/>

What's a GET request?

- Gets content from a page
- Can contain arguments / subpaths
 - `example.com/api?arg1=my_argument`
 - `example.com/api/my_argument`
- Use cases:
 - Display a web page on a browser
 - Access information on an API endpoint
- Examples:
 - Visiting <https://www.cs.rutgers.edu/>
 - Searching for a class on DegreeNav
 - *What are some more examples?*



Live example: <https://weblogin.cs.rutgers.edu/>

What's a POST request?

- **Posts** data at URL
- Data posted by client is handled according to server specification
 - Commonly JSON
- Use cases:
 - Posting comment (*upload comment to URL*)
 - Logging in (*submit username and password*)
 - Specific query on API

What's a POST request? (2)

- Examples:
 - Logging into DegreeNav with user/pass
 - Adding a course to your schedule
 - *What are some more examples?*

Test Your Understanding 1

Which of the following statements accurately describes the primary purpose of a GET request?

- a) GET requests are primarily used for submitting sensitive information such as passwords.
- b) GET requests are great for modifying and adding items to a database.
- c) GET requests are used to retrieve data from a server.
- d) GET requests are interchangeable with POST requests and can be used in all scenarios.

Test Your Understanding 1

Which of the following statements accurately describes the primary purpose of a GET request?

- a) GET requests are primarily used for submitting sensitive information such as passwords.
- b) GET requests are great for modifying and adding items to a database.
- c) GET requests are used to retrieve data from a server.**
- d) GET requests are interchangeable with POST requests and can be used in all scenarios.

Test Your Understanding 2

You go on Google and you type something into the search bar, then search - what type of request is happening?

- a. GET Requests
- b. POST Requests
- c. Neither
- d. Both GET and POST requests

Test Your Understanding 2

You go on Google and you type something into the search bar, then search - what type of request is happening?

- a. GET Requests
- b. POST Requests
- c. Neither
- d. Both GET and POST requests**

The background is a solid dark red color. It features several faint, stylized white outlines of mobile phones. One phone in the upper center displays a satellite dish icon. Another phone in the lower left shows a lightbulb icon. Various geometric shapes like circles and lines are scattered across the background, creating a tech-themed aesthetic.

Async Functions

Asynchronous Functions

When requesting data from anywhere, data does not come immediately... (there is a delay)

- This delay is a big no-no when working with *synchronous code*. (Everything you've written so far)
- Code waits for nobody.

Asynchronous Functions - 2

- How do we handle waiting for data?
 - Declare a **async** function.
- A async function declares a function to rely on a “Promise-based” behavior.
 - Enables the code to continue running without needing to wait for something to return.

JavaScript “async” and “await”

What are the following?

- Promise
- async
- await

Promise

“A Promise is a proxy for a value not necessarily known when the promise is created. It allows you to associate handlers with an asynchronous action's eventual success value or failure reason.” (mdn)

Imagine that you're a top singer, and fans ask day and night for your upcoming song.

To get some relief, you promise to send it to them when it's published. You give your fans a list. They can fill in their email addresses, so that when the song becomes available, all subscribed parties instantly receive it. And even if something goes very wrong, say, a fire in the studio, so that you can't publish the song, they will still be notified. (javascript.info)

Promise

“A Promise is a proxy for a value not necessarily known when the promise is created. It allows you to associate handlers with an asynchronous action's eventual success value or failure reason.” (mdn)

What is a promise?

- Signifies that something will be returned, whether it's successful or not
- Has three states: pending, fulfilled (success), rejected (failed)
- When completed, return

Why is this important?

Many operations are never instant

- database operations... (we will go over this later)
- practically anything over the internet

Promises allow us to handle things when they are completed.

JavaScript “await” and “async”

await - ‘await’ a promise. yield until promise is fulfilled or rejected

async - this decorator indicates that a function returns a promise

```
const revokeAccess = async () => {  
  var revoke = {  
    method: 'POST',  
    url: 'https://api.vault.netvoyage.com/v1/OAuth/revoke',  
    headers: {'content-type': 'application/x-www-form-urlencoded', 'Authorization': `Bearer ${access_token}`},  
    data: new URLSearchParams({  
      token: access_token  
    })  
  };  
  if(access_token) {  
    await axios.request(revoke).then(function (response) {  
      console.log(response.data);  
    }).catch(function (error) {  
      console.error(error);  
    });  
  } else {console.log("No access token found")}  
}
```



Working with Express

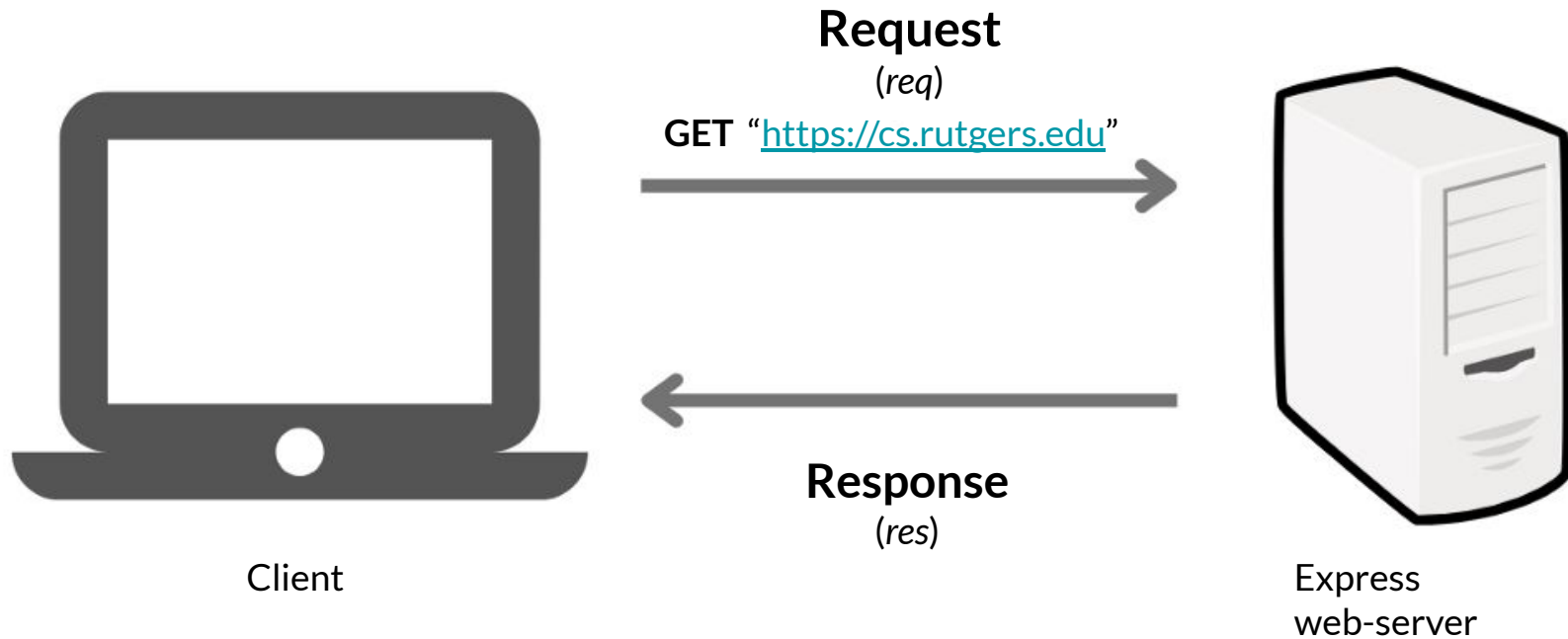
Handling GET requests

How do we handle users requesting data from URLs?

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```



Live example: <https://www.cs.rutgers.edu/>

Handling GET requests (2)

How else can we manage GET requests?

- paths / subpaths
- query strings
- parameters

Utilizing the **req** that Express gives us.

GET Requests Demo

```
const express = require('express')
const app = express()
const port = 3000

// Paths
app.get('/example_path', (req, res) => {
  res.send(`We've reached ./example_path!!`)
})

// Params
app.get('/api/:example_param', (req, res) => {
  res.send(`We are on a sample API endpoint. Our <b>param</b> is <b>${req.params.example_param}</b>!!`)
})

// Query string. e.g.: 0.0.0.0:3000/api?key=1234
app.get('/api/', (req, res) => {
  res.send(`We are on a sample API endpoint. Our <b>query</b> string is <b>${req.query.key}</b>!!`)
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Understanding GET requests

Where have you seen these used? How else can they be used?

The background is a solid dark red color. On the left side, there is a faint, stylized illustration of a smartphone. The phone's screen shows a satellite dish icon, and below the screen, there is a lightbulb icon. The phone is tilted diagonally.

POST requests

Handling POST requests

How do we handle users posting data to URLs?

```
...
const path = require('path')
...
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, './pages/post.html'));
})

app.post('/api/post', (req, res) =>e {
  res.send('POST request to the homepage' )
})

app.listen(port, () => {
  console.log(`Example app listening on port  ${port}`)
})
```

Handling POST requests (2.)

Live demonstration. Code can be found on [GitHub](#)

Understanding POST requests

Where have you seen these used? How else can they be used?

Questions?

Please fill out the feedback form when you have a chance!

Feedback Form



Next week...

- Introducing Supabase!
 - Experimenting with Supabase
 - Writing queries
 - CRUD operations