1. **What is Express.js?**
    - A) A database management system
    - B) A web server framework for Node.js
    - C) A front-end library
    - D) A CSS framework

    Answer: B) A web server framework for Node.js

2. **Which method is used to serve static files in Express.js?**
    - A) `app.get()`
    - B) `app.use()`
    - C) `app.static()`
    - D) `app.serve()`

    Answer: B) `app.use()`

3. **How do you include the middleware for serving static files in Express.js?**
    - A) `app.static('/public', 'public')`
    - B) `app.use(express.static('public'))`
    - C) `app.get('/public', express.static('public'))`
    - D) `app.middleware('/public', 'public')`

    Answer: B) `app.use(express.static('public'))`

4. **Which method is used to handle routing in Express.js?**
    - A) `app.route()`
    - B) `app.handle()`
    - C) `app.use()`
    - D) `app.get()`

    Answer: A) `app.route()`

5. **What is a route parameter in Express.js?**
    - A) A variable included in the URL path
    - B) A query parameter in the URL
    - C) An environment variable
    - D) A static file path

    Answer: A) A variable included in the URL path

6. **Which of the following is NOT a routing method in Express.js?**
    - A) `app.get()`
    - B) `app.post()`

- C) `app.put()`
- D) `app.delete()`
- E) `app.patch()`
- F) `app.create()`

Answer: F) `app.create()`

7. **What does `app.route('/users/:id')` represent?**
   - A) A static file path
   - B) A route with a dynamic parameter `id`
   - C) A route for serving static files
   - D) A database query

   Answer: B) A route with a dynamic parameter `id`

8. **What is a route handler in Express.js?**
   - A) A middleware function that handles requests for a specific route
   - B) A function that serves static files
   - C) A method to handle database connections
   - D) A function to parse request bodies

   Answer: A) A middleware function that handles requests for a specific route

9. **What is the purpose of `res.send()` in Express.js?**
   - A) To send a response to the client
   - B) To parse the request body
   - C) To set headers
   - D) To handle errors

   Answer: A) To send a response to the client

10. **Which method is used to send JSON responses in Express.js?**
    - A) `res.send()`
    - B) `res.json()`
    - C) `res.sendFile()`
    - D) `res.render()`

    Answer: B) `res.json()`

11. **What is middleware in Express.js?**
    - A) A tool for managing databases
    - B) Functions that execute during the request-response cycle
    - C) A framework for handling static files
    - D) A method for error logging

Answer: B) Functions that execute during the request-response cycle

12. **What is application-level middleware?**
    - A) Middleware applied to a specific router
    - B) Middleware applied globally to all routes in the app
    - C) Middleware that handles errors only
    - D) Middleware used for serving static files

Answer: B) Middleware applied globally to all routes in the app

13. **What is router-level middleware in Express.js?**
    - A) Middleware that applies to the entire application
    - B) Middleware that applies to a specific router or route
    - C) Middleware that handles request logging
    - D) Middleware that serves static files

Answer: B) Middleware that applies to a specific router or route

14. **How is error-handling middleware defined in Express.js?**
    - A) By using `app.use()`
    - B) By defining a function with four arguments: `err`, `req`, `res`, `next`
    - C) By using `app.error()`
    - D) By calling `next(err)`

Answer: B) By defining a function with four arguments: `err`, `req`, `res`, `next`

15. **Which of the following is an example of third-party middleware?**
    - A) `express.json()`
    - B) `body-parser`
    - C) `express.static()`
    - D) `express.Router()`

Answer: B) `body-parser`

16. **What does `next()` do in middleware functions?**
    - A) Ends the request-response cycle
    - B) Passes control to the next middleware function
    - C) Parses the request body
    - D) Sends a response to the client

Answer: B) Passes control to the next middleware function

17. **What is the lifecycle of middleware in Express.js?**
    - A) Middleware runs after the response is sent

- B) Middleware runs before the route handler is called
- C) Middleware runs during the response handling only
- D) Middleware is executed only once per application

Answer: B) Middleware runs before the route handler is called

18. **Which middleware function is used to parse incoming JSON payloads in Express.js?**
    - A) `express.urlencoded()`
    - B) `body-parser.json()`
    - C) `express.json()`
    - D) `body-parser.urlencoded()`

    Answer: C) `express.json()`

19. **What is the primary purpose of `express.urlencoded()` middleware?**
    - A) To parse JSON bodies
    - B) To parse URL-encoded bodies
    - C) To handle file uploads
    - D) To serve static files

    Answer: B) To parse URL-encoded bodies

20. **What is the difference between blocking and non-blocking code?**
    - A) Blocking code pauses execution until a task is complete, while non-blocking code continues execution
    - B) Non-blocking code pauses execution until a task is complete, while blocking code continues execution
    - C) Blocking code is always synchronous, while non-blocking code is always asynchronous
    - D) Blocking code cannot handle asynchronous operations

    Answer: A) Blocking code pauses execution until a task is complete, while non-blocking code continues execution

21. **How does a request travel through Express.js?**
    - A) It directly accesses the database
    - B) It goes through middleware functions and route handlers
    - C) It goes through static file servers
    - D) It is parsed directly by the client

    Answer: B) It goes through middleware functions and route handlers

22. **Which module is commonly used to parse request bodies in Express.js applications?**

- A) `express-body-parser`
- B) `body-parser`
- C) `express-request`
- D) `request-body-parser`

Answer: B) `body-parser`

23. **What does the `body-parser` middleware do?**
    - A) Handles static files
    - B) Parses incoming request bodies
    - C) Manages routing
    - D) Handles cookie parsing

    Answer: B) Parses incoming request bodies

24. **Which method would you use to parse a form submission with `application/x-www-form-urlencoded` content type?**
    - A) `bodyParser.json()`
    - B) `bodyParser.urlencoded({ extended: true })`
    - C) `bodyParser.text()`
    - D) `bodyParser.raw()`

    Answer: B) `bodyParser.urlencoded({ extended: true })`

25. **What is the effect of using `res.sendFile()`?**
    - A) Sends a JSON response to the client
    - B) Sends a file as a response to the client
    - C) Sends HTML content to the client
    - D) Sends an error response to the client

    Answer: B) Sends a file as a response to the client

26. **Which of the following is a method for sending a response with a specific status code in Express.js?**
    - A) `res.status(code).send()`
    - B) `res.set(code).send()`
    - C) `res.code(code).send()`
    - D) `res.send(code)`

    Answer: A) `res.status(code).send()`

27. **What does `res.redirect()` do in Express.js?**
    - A) Redirects the client to a different URL

- B) Sends a file to the client
- C) Sets a cookie
- D) Parses the incoming request body

Answer: A) Redirects the client to a different URL

28. **Which middleware function is typically used to handle JSON payloads in requests?**
    - A) `express.json()`
    - B) `express.urlencoded()`
    - C) `express.raw()`
    - D) `express.text()`

    Answer: A) `express.json()`

29. **What does `res.render()` do?**
    - A) Sends a file to the client
    - B) Sends HTML content to the client after processing a template
    - C) Redirects the client to a new URL
    - D) Sends a JSON response

    Answer: B) Sends HTML content to the client after processing a template

30. **Which method is used to set HTTP headers in the response?**
    - A) `res.set()`
    - B) `res.header()`
    - C) `res.headers()`
    - D) `res.headers.set()`

    Answer: A) `res.set()`

31. **What does `app.get('/users/:userId')` signify in routing?**
    - A) A route with a dynamic parameter `userId`
    - B) A static file route
    - C) A route that returns a JSON response
    - D) A middleware function

    Answer: A) A route with a dynamic parameter `userId`

32. **Which method would you use to handle HTTP POST requests?**
    - A) `app.get()`
    - B) `app.post()`
    - C) `app.put()`

- D) `app.delete()`

Answer: B) `app.post()`

33. **How do you handle multiple HTTP methods for the same route?**
    - A) By using `app.route()` and chaining methods
    - B) By creating separate routes for each method
    - C) By using `app.all()`
    - D) By combining methods in one function

    Answer: A) By using `app.route()` and chaining methods

34. **What is the purpose of `req.params` in Express.js?**
    - A) To access URL parameters
    - B) To parse request bodies
    - C) To access query parameters
    - D) To set headers

    Answer: A) To access URL parameters

35. **What does `app.param('id', callback)` do?**
    - A) It sets a middleware to handle route parameters
    - B) It parses the request body
    - C) It handles error responses
    - D) It sends a file to the client

    Answer: A) It sets a middleware to handle route parameters

36. **How can you specify a route that matches any path within a certain pattern?**
    - A) By using wildcards in route paths, such as `/users/*`
    - B) By using regular expressions in route paths
    - C) By using `app.use()` with a specific pattern
    - D) By setting default routes

    Answer: B) By using regular expressions in route paths

37. **What is the purpose of `app.all()` in Express.js?**
    - A) To define a middleware function for all HTTP methods
    - B) To handle specific HTTP methods
    - C) To set global middleware
    - D) To define a catch-all route

    Answer: A) To define a middleware function for all HTTP methods

38. **How do you define a route with multiple handlers in Express.js?**

- A) By chaining methods in `app.route()`
- B) By using `app.use()` with an array of handlers
- C) By defining multiple `app.get()` for the same route
- D) By defining a function that calls other functions

Answer: A) By chaining methods in `app.route()`

39. **Which method would you use to update a resource?**
    - A) `app.post()`
    - B) `app.get()`
    - C) `app.put()`
    - D) `app.delete()`

    Answer: C) `app.put()`

40. **How do you handle form submissions in Express.js?**
    - A) By using `express.urlencoded()` middleware
    - B) By using `express.json()` middleware
    - C) By using `express.raw()` middleware
    - D) By using `express.text()` middleware

    Answer: A) By using `express.urlencoded()` middleware

41. **What does error-handling middleware typically look like?**
    - A) `function (err, req, res, next) { /* error handling logic */ }`
    - B) `function (req, res, next) { /* error handling logic */ }`
    - C) `function (err, req, res) { /* error handling logic */ }`
    - D) `function (req, res) { /* error handling logic */ }`

    Answer: A) `function (err, req, res, next) { /* error handling logic */ }`

42. **What is the role of third-party middleware in Express.js?**
    - A) To provide additional functionality not included in Express.js core
    - B) To handle static files
    - C) To manage internal application logic
    - D) To define route handlers

    Answer: A) To provide additional functionality not included in Express.js core

43. **What is the primary use of `express.Router()`?**
    - A) To create modular route handlers

- B) To handle middleware functions
- C) To parse request bodies
- D) To serve static files

Answer: A) To create modular route handlers

44. **How does middleware affect the request-response cycle in Express.js?**
    - A) Middleware runs before the request is processed by the route handler
    - B) Middleware runs only after the response is sent
    - C) Middleware runs after the route handler
    - D) Middleware handles static file requests

    Answer: A) Middleware runs before the request is processed by the route handler

45. **Which of the following is true about the `next()` function in Express.js middleware?**
    - A) It must be called to pass control to the next middleware or route handler
    - B) It ends the request-response cycle
    - C) It is used to send responses to the client
    - D) It is used to set HTTP headers

    Answer: A) It must be called to pass control to the next middleware or route handler

46. **How do you create a middleware function in Express.js?**
    - A) By defining a function that takes `req`, `res`, and `next` as parameters
    - B) By defining a function that takes `req` and `res` as parameters
    - C) By using `app.use()` without parameters
    - D) By using `app.use()` with a callback function

    Answer: A) By defining a function that takes `req`, `res`, and `next` as parameters

47. **What does `app.use()` do in Express.js?**
    - A) Registers middleware functions
    - B) Defines route handlers
    - C) Sends responses to the client
    - D) Parses request bodies

    Answer: A) Registers middleware functions

48. **When would you use `app.use()` with a specific path?**
    - A) To apply middleware to a subset of routes
    - B) To define a global middleware
    - C) To handle error responses
    - D) To parse URL parameters

Answer: A) To apply middleware to a subset of routes

49. **How do you define a middleware function that only applies to routes under `/api`?**
    - A) `app.use('/api', apiMiddleware)`
    - B) `app.use(apiMiddleware, '/api')`
    - C) `app.use('/api', function(req, res, next) { /* middleware logic */ })`
    - D) `app.use(function(req, res, next) { /* middleware logic */ }, '/api')`

    Answer: A) `app.use('/api', apiMiddleware)`

50. **What is the difference between application-level and router-level middleware?**
    - A) Application-level middleware applies globally, while router-level middleware applies to specific routers
    - B) Router-level middleware is only used for error handling
    - C) Application-level middleware is used to parse request bodies
    - D) Router-level middleware is used to serve static files

    Answer: A) Application-level middleware applies globally, while router-level middleware applies to specific routers

51. **Which type of code allows other operations to continue while waiting for a task to complete?**
    - A) Blocking code
    - B) Non-blocking code
    - C) Synchronous code
    - D) Sequential code

    Answer: B) Non-blocking code

52. **Which type of code halts execution until the current operation finishes?**
    - A) Non-blocking code
    - B) Blocking code
    - C) Asynchronous code
    - D) Concurrent code

    Answer: B) Blocking code

53. **What is a common example of non-blocking code in Node.js?**
    - A) Synchronous file reading
    - B) Asynchronous file reading using `fs.readFile()`
    - C) Synchronous HTTP requests
    - D) Sequential execution of functions

Answer: B) Asynchronous file reading using `fs.readFile()`

54. **Why is non-blocking code advantageous in a web server environment?**
    - A) It allows handling multiple requests concurrently without waiting for each to finish
    - B) It ensures that operations are executed in sequence
    - C) It simplifies code management
    - D) It automatically handles errors

    Answer: A) It allows handling multiple requests concurrently without waiting for each to finish

55. **How can you perform asynchronous operations in Node.js?**
    - A) By using callbacks
    - B) By using promises
    - C) By using async/await syntax
    - D) All of the above

    Answer: D) All of the above

56. **What is a callback in Node.js?**
    - A) A function passed as an argument to be executed after an operation completes
    - B) A function that handles HTTP requests
    - C) A method for serving static files
    - D) A way to parse incoming request bodies

    Answer: A) A function passed as an argument to be executed after an operation completes

57. **What does the `fs.readFile()` function do in Node.js?**
    - A) Reads a file synchronously
    - B) Reads a file asynchronously
    - C) Writes data to a file
    - D) Deletes a file

    Answer: B) Reads a file asynchronously

58. **Which of the following is an advantage of asynchronous code?**
    - A) It reduces the need for multiple threads
    - B) It increases the complexity of code
    - C) It makes the code easier to understand
    - D) It halts execution until a task is complete

    Answer: A) It reduces the need for multiple threads

59. **How can you handle asynchronous code using promises?**
   - A) By chaining `.then()` and `.catch()` methods
   - B) By using `async` and `await` keywords
   - C) By using callbacks
   - D) By using synchronous methods

   Answer: A) By chaining `.then()` and `.catch()` methods

60. **What is the purpose of `async` and `await` keywords in JavaScript?**
   - A) To handle asynchronous operations more easily
   - B) To perform synchronous operations
   - C) To manage static file serving
   - D) To handle routing in Express.js

   Answer: A) To handle asynchronous operations more easily

61. **Which of the following is NOT a core feature of Express.js?**
   - A) Routing
   - B) Middleware support
   - C) Database management
   - D) Static file serving

   Answer: C) Database management

62. **What is the role of `express.Router()` in creating modular route handlers?**
   - A) It helps organize routes and middleware into separate modules
   - B) It parses request bodies
   - C) It handles error responses
   - D) It manages HTTP headers

   Answer: A) It helps organize routes and middleware into separate modules

63. **Which method allows you to serve an HTML file as a response in Express.js?**
   - A) `res.sendFile()`
   - B) `res.send()`
   - C) `res.render()`
   - D) `res.json()`

   Answer: A) `res.sendFile()`

64. **How do you use middleware to handle errors in Express.js?**
   - A) By defining a middleware function with four arguments: `err`, `req`, `res`, `next`
   - B) By using `app.use()` without arguments
   - C) By calling `next(err)` in route handlers

- D) By using `app.get()` for error handling

Answer: A) By defining a middleware function with four arguments: `err`, `req`, `res`, `next`

65. **What is the purpose of the `req.body` object in Express.js?**
    - A) To access the parsed body of a request
    - B) To handle routing parameters
    - C) To manage static files
    - D) To set response headers

Answer: A) To access the parsed body of a request

66. **What does `app.all()` do when used without a path argument?**
    - A) Applies middleware to all HTTP methods for all paths
    - B) Defines a catch-all route for all HTTP methods
    - C) Handles only GET requests
    - D) Handles error responses

Answer: A) Applies middleware to all HTTP methods for all paths

67. **Which middleware function parses incoming requests with JSON payloads?**
    - A) `express.urlencoded()`
    - B) `express.json()`
    - C) `body-parser.urlencoded()`
    - D) `body-parser.json()`

Answer: B) `express.json()`

68. **What is the purpose of the `req.query` object in Express.js?**
    - A) To access query string parameters in the URL
    - B) To access URL path parameters
    - C) To handle form submissions
    - D) To manage request headers

Answer: A) To access query string parameters in the URL

69. **Which of the following methods is used to set a cookie in Express.js?**
    - A) `res.setCookie()`
    - B) `res.cookie()`
    - C) `res.set()`
    - D) `res.cookieSet()`

Answer: B) `res.cookie()`

70. **How do you handle file uploads in Express.js?**
   - A) By using third-party middleware like `multer`
   - B) By using built-in Express.js functions
   - C) By using `body-parser`
   - D) By using `express.static()`

   Answer: A) By using third-party middleware like `multer`

71. **What does `app.use(express.json())` do in an Express.js application?**
   - A) It parses incoming JSON requests and puts the parsed data in `req.body`
   - B) It handles static file requests
   - C) It sets global headers for all responses
   - D) It manages routing

   Answer: A) It parses incoming JSON requests and puts the parsed data in `req.body`

72. **Which method is used to respond with HTML content in Express.js?**
   - A) `res.send()`
   - B) `res.json()`
   - C) `res.redirect()`
   - D) `res.render()`

   Answer: A) `res.send()`

73. **What does `app.use()` with a path argument do?**
   - A) It applies middleware only to routes that match the specified path
   - B) It handles errors globally
   - C) It sets up a route for handling requests
   - D) It defines static file paths

   Answer: A) It applies middleware only to routes that match the specified path

74. **How can you debug middleware functions in Express.js?**
   - A) By adding `console.log()` statements in the middleware functions
   - B) By using `app.debug()`
   - C) By inspecting `req` and `res` objects in route handlers
   - D) By using `res.send()` to check middleware execution

   Answer: A) By adding `console.log()` statements in the middleware functions

75. **What is a common use case for `app.use()` with a path argument in Express.js?**
   - A) To apply middleware only to routes under a specific path, such as `/api`
   - B) To define static file serving

  - C) To set global request headers
  - D) To manage routing for specific HTTP methods

Answer: A) To apply middleware only to routes under a specific path, such as `/api`

76. **What is the function of `req.method` in an Express.js route handler?**
    - A) It provides the HTTP method of the request (e.g., GET, POST)
    - B) It returns the URL of the request
    - C) It parses query parameters
    - D) It sets the request headers

Answer: A) It provides the HTTP method of the request (e.g., GET, POST)

77. **What does `res.status(404).send('Not Found')` do?**
    - A) Sends a 404 status code with a 'Not Found' message
    - B) Redirects the client to a 404 page
    - C) Sets a 404 status code without sending a response
    - D) Sets a 'Not Found' error in the request object

Answer: A) Sends a 404 status code with a 'Not Found' message

78. **How do you ensure that middleware runs only for specific routes in Express.js?**
    - A) By specifying the route path in `app.use()` or `router.use()`
    - B) By using global middleware functions
    - C) By defining middleware functions in route handlers
    - D) By using `app.all()` with specific paths

Answer: A) By specifying the route path in `app.use()` or `router.use()`

79. **Which method is used to respond with a file download in Express.js?**
    - A) `res.download()`
    - B) `res.file()`
    - C) `res.sendFile()`
    - D) `res.attachment()`

Answer: A) `res.download()`

80. **What does `res.redirect('/home')` do in an Express.js route handler?**
    - A) It sends a redirect response to the client, directing it to `/home`
    - B) It serves the `/home` file from the file system
    - C) It renders the `/home` template
    - D) It sets up a new route for `/home`

Answer: A) It sends a redirect response to the client, directing it to `/home`

Additional Questions

1. **What does `express()` do in an Express.js application?**
    - A) It creates a new middleware function
    - B) It initializes an Express application
    - C) It starts the server
    - D) It serves static files

   Answer: B) It initializes an Express application

2. **Which method is used to start an Express server and listen on a specified port?**
    - A) `app.listen()`
    - B) `app.start()`
    - C) `app.init()`
    - D) `app.run()`

   Answer: A) `app.listen()`

3. **How do you set a port for an Express application to listen on?**
    - A) `app.listen(port)`
    - B) `app.set('port', port)`
    - C) `app.port(port)`
    - D) `app.use(port)`

   Answer: A) `app.listen(port)`

4. **What middleware function parses incoming request bodies in Express.js?**
    - A) `express.bodyParser()`
    - B) `express.json()`
    - C) `express.urlencoded()`
    - D) `express.parser()`

   Answer: B) `express.json()`

5. **Which Express method is used to handle HTTP POST requests?**
    - A) `app.post()`
    - B) `app.get()`
    - C) `app.put()`
    - D) `app.delete()`

   Answer: A) `app.post()`

6. **How do you serve static files from a directory named 'public' in an Express.js application?**
   - A) `app.use(express.static('public'))`
   - B) `app.static('public')`
   - C) `app.serve('public')`
   - D) `app.use('/public', express.static('public'))`

   Answer: A) `app.use(express.static('public'))`

7. **Which method sends a file as an HTTP response in Express.js?**
   - A) `res.sendFile()`
   - B) `res.file()`
   - C) `res.download()`
   - D) `res.serveFile()`

   Answer: A) `res.sendFile()`

8. **To serve static files under the '/assets' route, which code snippet should you use?**
   - A) `app.use('/assets', express.static('assets'))`
   - B) `app.use(express.static('/assets'))`
   - C) `app.static('/assets', 'assets')`
   - D) `app.use('/assets', express.static('public'))`

   Answer: A) `app.use('/assets', express.static('assets'))`

9. **What happens if a static file and a route handler match the same URL?**
   - A) The static file is served, and the route handler is ignored.
   - B) The route handler is executed, and the static file is ignored.
   - C) Both the static file and route handler are executed.
   - D) An error is thrown.

   Answer: B) The route handler is executed, and the static file is ignored.

10. **Which of the following is NOT a valid static file type served by Express?**
    - A) HTML files
    - B) CSS files
    - C) JavaScript files
    - D) Database queries

    Answer: D) Database queries

11. **Which method is used to handle GET requests to a specific path in Express.js?**
    - A) `app.post()`

    - B) `app.get()`
    - C) `app.put()`
    - D) `app.all()`

    Answer: B) `app.get()`

12. **How do you define a route that will handle all HTTP methods at a specific path?**
    - A) `app.route()`
    - B) `app.all()`
    - C) `app.use()`
    - D) `app.method()`

    Answer: B) `app.all()`

13. **Which Express method allows defining multiple routes at once for different HTTP methods?**
    - A) `app.method()`
    - B) `app.route()`
    - C) `app.all()`
    - D) `app.use()`

    Answer: B) `app.route()`

14. **How do you define a route handler for POST requests to the '/login' path?**
    - A) `app.post('/login', handler)`
    - B) `app.get('/login', handler)`
    - C) `app.use('/login', handler)`
    - D) `app.all('/login', handler)`

    Answer: A) `app.post('/login', handler)`

15. **Which method is used to handle DELETE requests in Express.js?**
    - A) `app.get()`
    - B) `app.put()`
    - C) `app.delete()`
    - D) `app.post()`

    Answer: C) `app.delete()`

16. **What is the purpose of a wildcard route parameter in Express.js?**
    - A) To match a specific URL path
    - B) To match any route path segment

- C) To specify a query string
- D) To define static file paths

Answer: B) To match any route path segment

17. **How do you define a route with an optional parameter?**
    - A) `/path/:param?`
    - B) `/path/:param*`
    - C) `/path/:param+`
    - D) `/path/:param/`

    Answer: A) `/path/:param?`

18. **Which Express.js feature allows matching routes with multiple path segments?**
    - A) `*` (Wildcard)
    - B) `?` (Optional)
    - C) `+` (One or more)
    - D) `:` (Dynamic parameters)

    Answer: A) `*` (Wildcard)

19. **How can you define a route that accepts multiple parameters in Express.js?**
    - A) `/route/:param1/:param2`
    - B) `/route/:param1?/:param2?`
    - C) `/route/:param1/*`
    - D) `/route/*`

    Answer: A) `/route/:param1/:param2`

20. **What will `req.params.id` contain if the route is `/item/:id` and the URL is `/item/456`?**
    - A) `456`
    - B) `/item/456`
    - C) `undefined`
    - D) `null`

    Answer: A) `456`