1 . Routing :

Routing refers to how an application's endpoints (URIs) respond to client requests.

** These routing methods specify a callback function (sometimes called "handler functions") called when the application receives a request to the specified route (endpoint) and HTTP method. In other words, the application "listens" for requests that match the specified route(s) and method(s), and when it detects a match, it calls the specified callback function.

** In fact, the routing methods can have more than one callback function as arguments. With multiple callback functions, it is important to provide `next` as an argument to the callback function and then call `next()` within the body of the function to hand off control to the next callback.

Example code :

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

// respond with "hello world" when a GET request is made to the homepage
app.get('/', (req, res) => {
  res.send('hello world')
})
```

Route Methods :

A route method is derived from one of the HTTP methods, and is attached to an instance of the `express` class.

```
// GET method route
app.get('/', (req, res) => {
  res.send('GET request to the homepage')
})

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

There is a special routing method, `app.all()`, used to load middleware functions at a path for ***all*** HTTP request methods.

For example :

```
app.all('/secret', (req, res, next) => {
  console.log('Accessing the secret section ...')
  next() // pass control to the next handler
})
```

Route Paths :

Route paths, in combination with a request method, define the endpoints at which requests can be made. Route paths can be strings, string patterns, or regular expressions.

** If you need to use the dollar character (`$`) in a path string, enclose it escaped within `([` and `])`.

For example, the path string for requests at "`/data/$book`", would be "`/data/([\$])book`".

This route path will match requests to `/about`.

```
app.get('/about', (req, res) => {
  res.send('about')
})
```

This route path will match requests to `/random.text`.

```
app.get('/random.text', (req, res) => {
  res.send('random.text')
})
```

This route path will match `acd` and `abcd`.

```
app.get('/ab?cd', (req, res) => {
  res.send('ab?cd')
})
```

Route Parameters :

Route parameters are named URL segments that are used to capture the values specified at their position in the URL. The captured values are populated in the `req.params` object, with the name of the route parameter specified in the path as their respective keys.

Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }

```
app.get('/users/:userId/books/:bookId', (req, res) => {
  res.send(req.params)
})
```

** Since the hyphen (`-`) and the dot (`.`) are interpreted literally, they can be used along with route parameters for useful purposes.

Route path: /flights/:from-:to
Request URL: http://localhost:3000/flights/LAX-SFO
req.params: { "from": "LAX", "to": "SFO" }

Route path: /plantae/:genus.:species
Request URL: http://localhost:3000/plantae/Prunus.persica
req.params: { "genus": "Prunus", "species": "persica" }

** To have more control over the exact string that can be matched by a route parameter, you can append a regular expression in parentheses (`( )`):

Route path: /user/:userId(\d+)
Request URL: http://localhost:3000/user/42
req.params: {"userId": "42"}

Route Handlers

Route handlers can be in the form of a function, an array of functions, or combinations of both, as shown in the following examples.

A single callback function can handle a route. For example:

```
app.get('/example/a', (req, res) => {
```

```
  res.send('Hello from A!')
})
```

** More than one callback function can handle a route (make sure you specify the `next` object). For example:

```
app.get('/example/b', (req, res, next) => {
  console.log('the response will be sent by the next function ...')
  next()
}, (req, res) => {
  res.send('Hello from B!')
})
```

An array of callback functions can handle a route. For example:

```
const cb0 = function (req, res, next) {
  console.log('CB0')
  next()
}

const cb1 = function (req, res, next) {
  console.log('CB1')
  next()
}

const cb2 = function (req, res) {
  res.send('Hello from C!')
}

app.get('/example/c', [cb0, cb1, cb2])
```

Response Method :

The methods on the response object (`res`) in the following table can send a response to the client, and terminate the request-response cycle. If none of these methods are called from a route handler, the client request will be left hanging.

Method	Description

res.download()	Prompt a file to be downloaded.
res.end()	End the response process.
res.json()	Send a JSON response.
res.jsonp()	Send a JSON response with JSONP support.
res.redirect()	Redirect a request.
res.render()	Render a view template.
res.send()	Send a response of various types.
res.sendFile()	Send a file as an octet stream.
res.sendStatus()	Set the response status code and send its string representation as the response body.

app.route()

You can create chainable route handlers for a route path by using `app.route()`. Because the path is specified at a single location,

```
app.route('/book')
  .get((req, res) => {
    res.send('Get a random book')
  })
  .post((req, res) => {
```

```
    res.send('Add a book')
  })
  .put((req, res) => {
    res.send('Update the book')
  })
```

express.Router

Use the `express.Router` class to create modular, mountable route handlers. A `Router` instance is a complete middleware and routing system; for this reason, it is often referred to as a “mini-app”.

```
const express = require('express')
const router = express.Router()

// middleware that is specific to this router
const timeLog = (req, res, next) => {
  console.log('Time: ', Date.now())
  next()
}
router.use(timeLog)

// define the home page route
router.get('/', (req, res) => {
  res.send('Birds home page')
})
// define the about route
router.get('/about', (req, res) => {
  res.send('About birds')
})

module.exports = router
```