# Responding to Requests

This lesson covers methods for responding to requests in a web server.

**WE'LL COVER THE FOLLOWING**

- Accessing Express Services
- Defining Routes
- Listening to Requests

The main job of a web server is to respond to HTTP requests. Here's the JavaScript code for a minimal Express-based web server that returns "Hello from Express!" for a request to the root URL.

```
// Load the Express package as a module
const express = require("express");

// Access the exported service
const app = express();

// Return a string for requests to the root URL ("/")
app.get("/", (request, response) => {
  response.send("Hello from Express!");
});

// Start listening to incoming requests
// If process.env.PORT is not defined, port number 3000 is used
const listener = app.listen(process.env.PORT || 3000, () => {
  console.log(`Your app is listening on port ${listener.address().port}`);
});
```

You can launch your server with either `node index.js` or `npm start`, then type its root URL (http://localhost:3000 if your server runs on your local machine) in a browser. You should see the string `"Hello from Express!"` appear. Let's dissect this example.

## Accessing Express Services #

Once Express is installed, you can load its package in your main application file and access the exported services provided by the framework. The beginning of the server code does just that.

```
// Load the Express package as a module
const express = require("express");

// Access the main Express object
const app = express();
```

## Defining Routes #

In web development terminology, a route is an entry point into an application. It is relative to the application URL. The `"/"` route matches the root of the application.

```
// Return a string for requests to the root URL ("/")
app.get("/", (request, response) => {
  response.send("Hello from Express!");
});
```

When an HTTP request is made to the route URL, the associated callback function is executed. This function takes as parameters objects representing the HTTP request and response. Here, the function body sends a text response with the content `"Hello from Express!"`.

## Listening to Requests #

To process incoming request, a web server must listen on a specific port. A port is a communication endpoint on a machine. The main Express object has a `listen()` method that takes as parameter the listening port and a callback function called for each request. The last part of the server code calls this method to start listening.

```
// Start listening to incoming requests
// If process.env.PORT is not defined, 3000 is used

const listener = app.listen(process.env.PORT || 3000, () => {
  console.log(`Your app is listening on port ${listener.address().port}`);
});
```