

Express 1

What is Express.js?

- Gives you access to a library of code (Node.js functions) that makes it easier for you to create web servers with Node
- Although Express is minimalist, developers have created compatible middleware packages to address almost any web development problem
- Unlike other frameworks, like Django, Express is an un-opinionated web framework



EXPRESS IS USED WITH

THE MEAN STACK



THE NERD STACK



COMPANIES WHICH USE EXPRESS



>
accenture

Uber



Why Use Express?

- To speed up development and decrease the amount of boilerplate code you have to write.
- We could write all the code we need without Express, but we will use Express to speed up development.

أكاديمية طويق
TUWAIQ ACADEMY



Install Express

- Express is another package that you need to install using NPM.
- Step 1: Open your terminal or command prompt.
- Step 2: Create a directory using the mkdir command and then navigate into it using the cd command.

```
mkdir myapp
```

```
cd myapp
```

Install Express

- Step 3: Use the npm init command to create a package.json file for your application. The command, npm init, will prompt you to enter several details, such as the package name, version, description, entry point, test command, etc.

`npm init`

- Step 4: To display the package.json file enter `cat package.json` or type `package.json`

Install Express

- Step 5: Install the Express library in the 'myapp' directory that you created and save it in the dependencies list of the package.json file:

`npm install express`

- Step 6: Enter `cat package.json` or type `package.json` into your terminal again. The dependencies section of your package.json will now appear in the package.json file and will contain Express.




Install Express

```
cat package.json
```

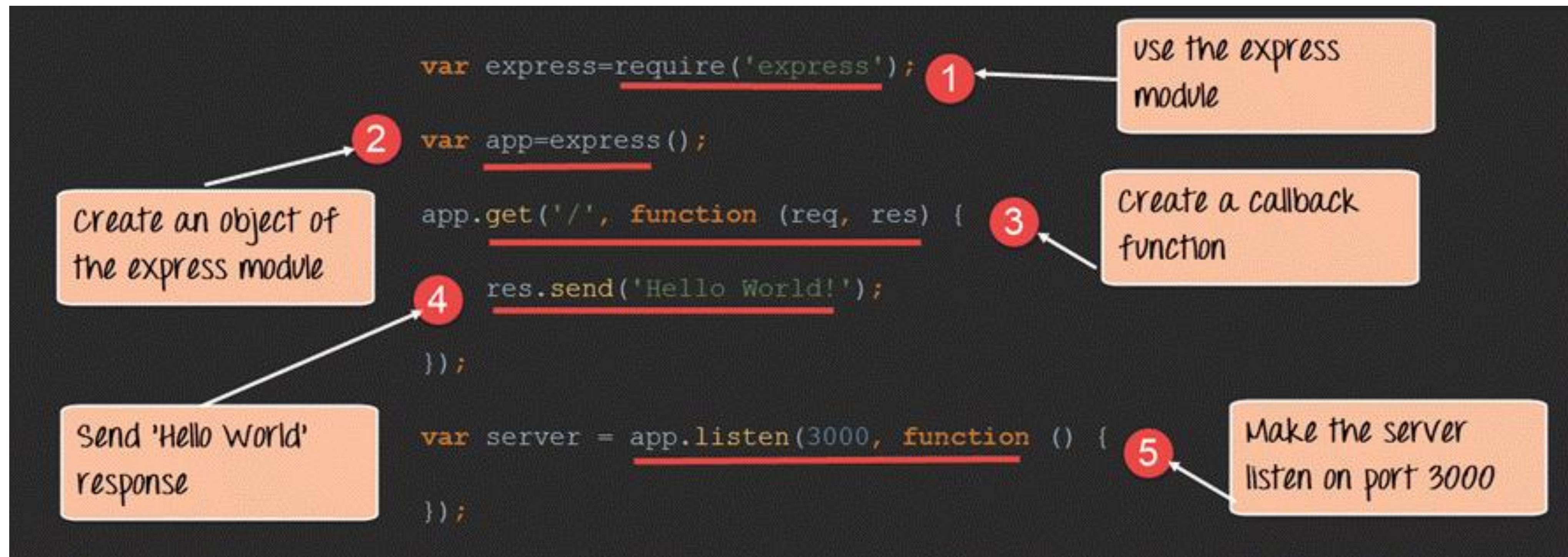
```
{  
  "name": "myapp",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "express": "^4.15.4"  
  }  
}
```


Restaurant analogy to explain four key parts of Express app

- The require statements
- Middleware
- Routing
- App.listen()/ Starting the server

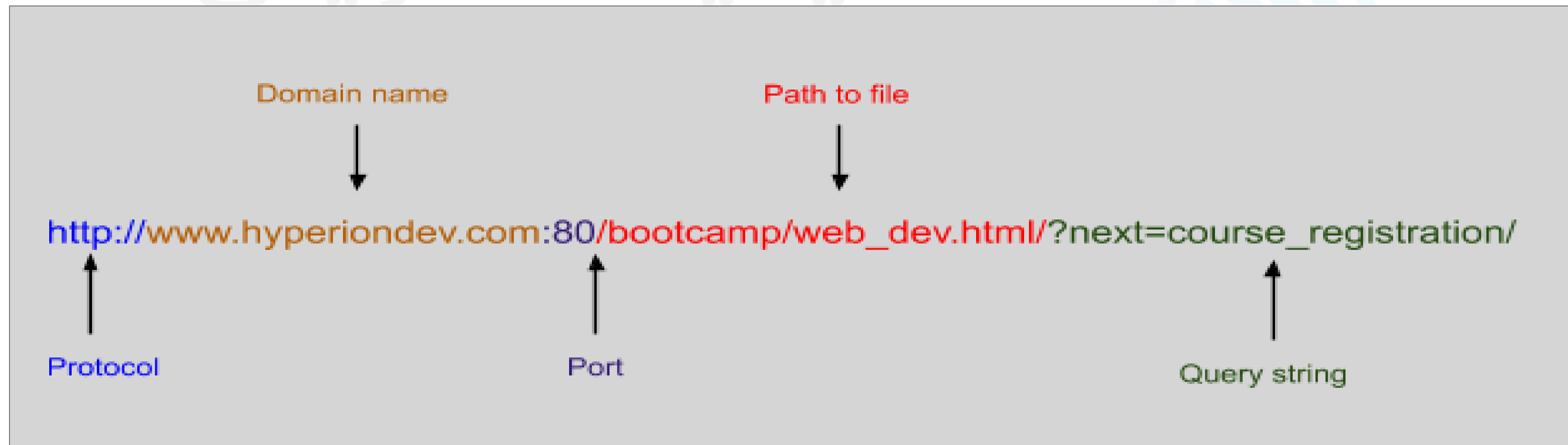
1	<code>const express = require('express' 4.16.2)</code>]		1- Hiring the manager
2	<code>const app = express()</code>			
3				
4	<code>app.use(function (req, res, next) {</code>			
5	<code> console.log('Request: ', req)</code>			
6	<code> console.log('Response: ', res)</code>]		2-Got shirt and shoes?
7	<code> next()</code>			
8	<code>})</code>			
9				
10	<code>app.get('/', function (req, res) {</code>			
11	<code> res.send('Hello World!')</code>]		3-Taking an order
12	<code>})</code>			
13				
14	<code>app.listen(3000, function () {</code>			
15	<code> console.log('Example app listening on port 3000!')</code>			
16	<code>})</code>			4-Open for business

Restaurant analogy to explain four key parts of Express app



Routing

- Routing: determining how an application responds to a client's request to a particular endpoint. The request is made using a URI (or path) and a specific HTTP request method.
- Remember that a URL contains a lot of information:



Routing

- To perform routing, we are interested in the path section of the URL
- With Express there are a few route methods used to perform routing: get, post, put and delete (we will use get):

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


Creating a Server Using Express

- In the root directory of your 'myapp' directory (the directory you created when you installed Express), create a file called 'app.js' and copy the code below into it:

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

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

app.listen(8000, function () {
  console.log('Example app listening on port 8000!')
})
```

Creating a Server Using Express

- To start the server, call node with the script in your terminal or command prompt

```
ress I>node app.js  
Example app listening on port 8000!
```

- Now use your web browser to navigate to <http://127.0.0.1:8000/>. You should see the string “Hello World!” displayed in your browser!

Serving Static Files with Express

- With Express it is easy to serve static resources by using the `express.static` built-in middleware function
- To allow your app to serve static files:
 1. Create a folder in your project directory to store the static files you want to serve.
 2. Add all the static resources that you want your app to make available (images, HTML files etc) to this folder.
 3. Add the following code to your `app.js` file: `app.use(express.static('public'));`

Environment Variables

- When creating back-end apps, it is important to be able to access environment variables.
- Node.js allows us to access these variables using **process.env**.
- To see some of the environment variables stored on your PC, add the following line of code to your app.js file and run it in the terminal:

```
console.log('The value of process.env is:', process.env);
```


Environment Variables

- An important environment variable that will be set on the server is the port number on which your application server will listen for HTTP requests.
- To get the port number from the environment variables instead of hardcoding it, we use the following code:

```
const PORT = process.env.PORT || 8000;  
app.listen(PORT, () => {  
  console.log(`Server is listening on port ${PORT}`);  
});
```

Nodemon

- Use Nodemon to enable server restarts on file changes.
- After typing the command shown below into the command line interface, you should see that your package.json file now includes a reference to Nodemon in its devDependencies section.

`npm install --save-dev nodemon`

- To run an app using Nodemon, type `nodemon name_of_file` in the terminal instead of `node name_of_file`.

Add a Start Script

- It is recommended that you add a script to your package.json file for every application that you create that specifies how to start your app.

```
"main": "app.js",  
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "start": "nodemon app.js"  
},
```

- To start the application, we then type **npm start**. This will use the instruction specified in the start script in the package.json file to run the code.



Resources

- <https://www.freecodecamp.org/news/going-out-to-eat-and-understanding-the-basics-of-express-js-f034a029fb66/>
- <https://www.coursereport.com/blog/what-is-express>
- <https://www.guru99.com/node-js-express.html>