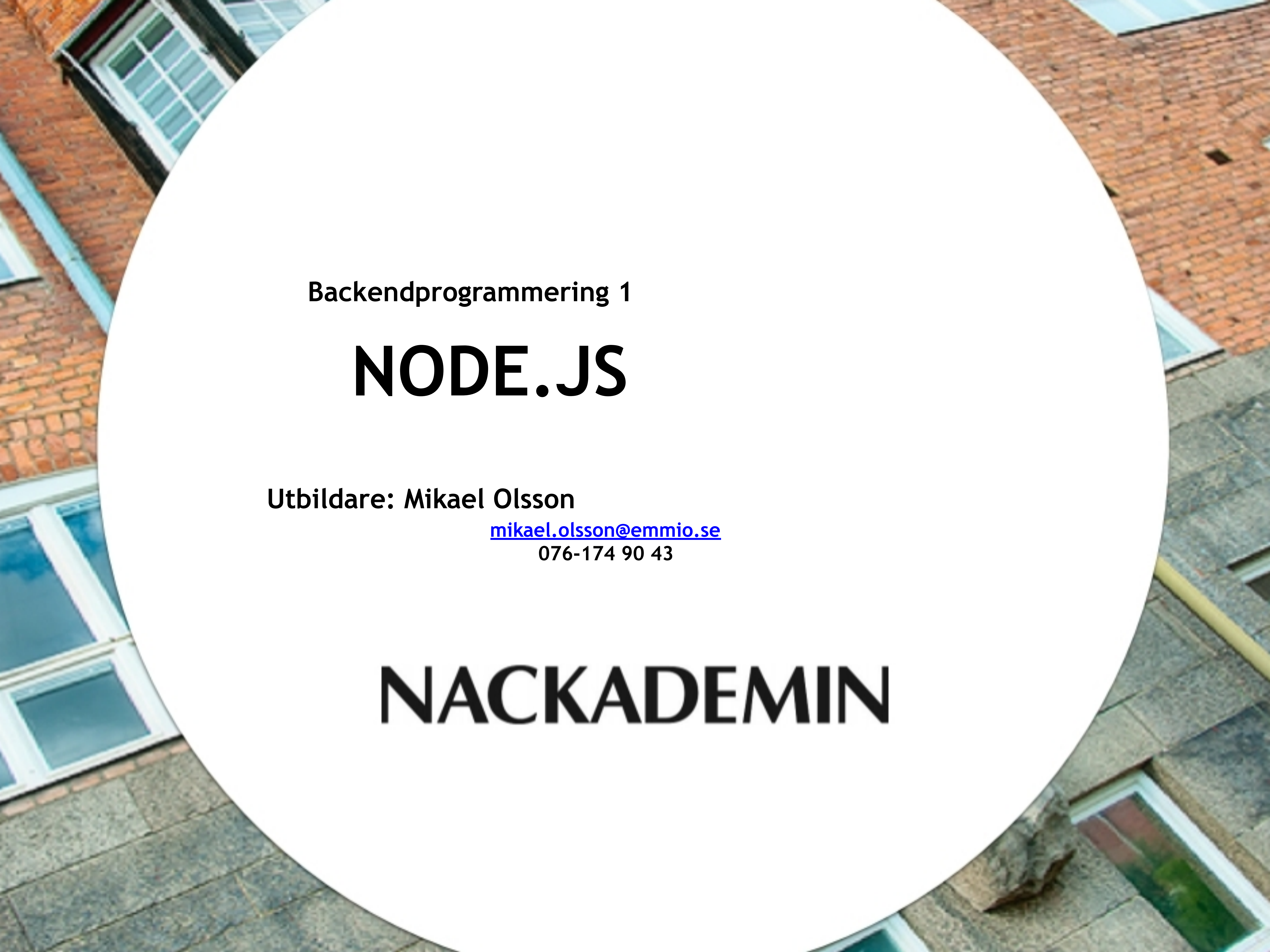


Backendprogrammering 1

NODE.JS

Utbildare: Mikael Olsson
mikael.olsson@emmio.se
076-174 90 43

NACKADEMIN

Skapa HTTP Webbserver i Node.js

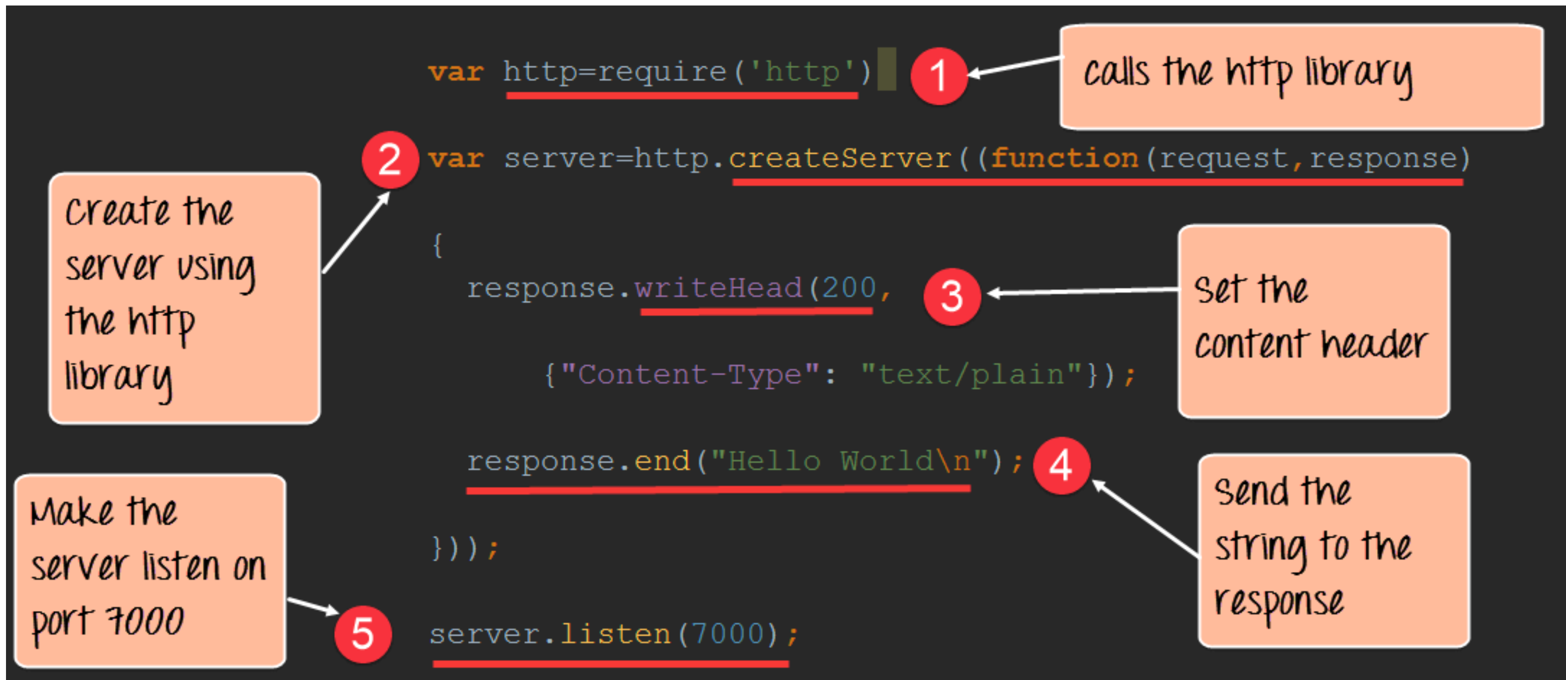
1. The basic functionality of the require function is that it reads a JavaScript file, executes the file, and then proceeds to return the exports object. So in our case, since we want to use the functionality of the http module, we use the require function to get the desired functions from the http module so that it can be used in our application.

2. In this line of code, we are creating a server application which is based on a simple function. This function is called whenever a request is made to our server application.

3. When a request is received, we are saying to send a response with a header type of '200.' This number is the normal response which is sent in an http header when a successful response is sent to the client.

4. In the response itself, we are sending the string 'Hello World.'

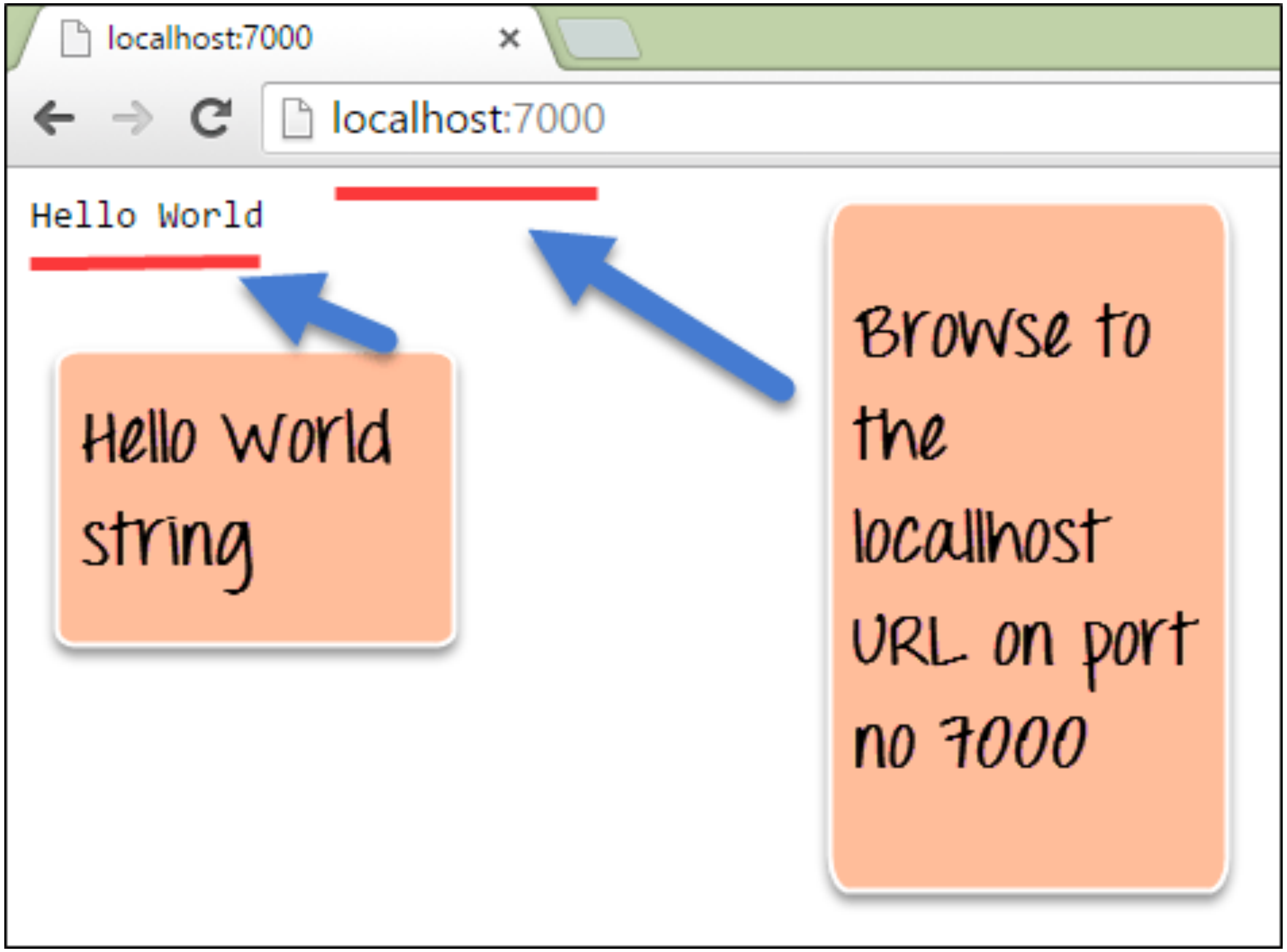
5. We are then using the server.listen function to make our server application listen to client requests on port no 7000. You can specify any available port over here.



```
var http=require('http')
var server=http.createServer((function(request,response)
{
  response.writeHead(200,
      {"Content-Type": "text/plain"});
  response.end("Hello World\n");
}));
server.listen(7000);
```

1
calls the http library
2
Create the server using the http library
3
Set the content header
4
Send the string to the response
5
Make the server listen on port 7000

Webbserver



localhost:7000

localhost:7000

```
Hello World
```

Göra GET-requests i Node.js

1.We are using the 'require' module which was installed in the last step. This module has the necessary functions which can be used to make GET requests to websites.

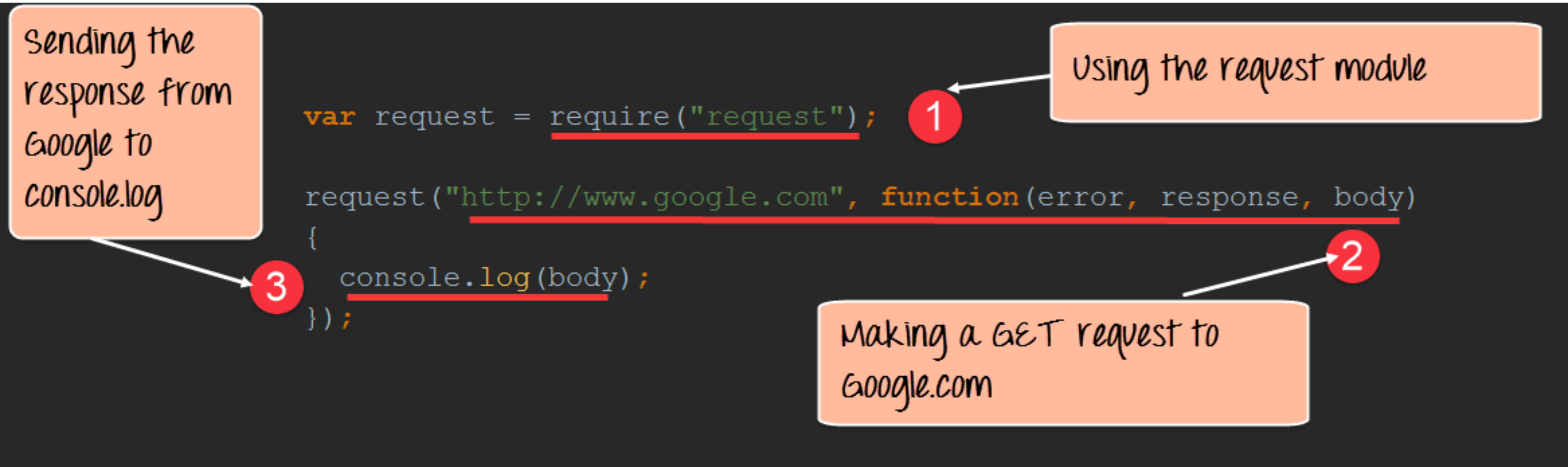
2.We are making a GET Request to www.google.com and subsequently calling a function when a response is received. When a response is received the parameters(error, response, and body) will have the following values

2.1.Error – In case there is any error received when using the GET request, it will be recorded here.

2.2.Response- The response will have the http headers which are sent back in the response.

2.3.Body- The body will contain the entire content of the response sent by Google.

3.In this, we are just writing the content received in the body parameter to the console.log file. So basically, whatever we get by going to www.google.com will be written to the console.log.



```
var request = require("request");

request("http://www.google.com", function(error, response, body)
{
  console.log(body);
});
```

Sending the response from Google to console.log
3
1
Using the request module
2
Making a GET request to Google.com

Stackar

- In computing, a solution stack or software stack is a set of software subsystems or components needed to create a complete platform such that no additional software is needed to support applications.
  - Wikipedia
- En vanlig variant är LAMP
  - Linux (operating system)
  - Apache (web server)
  - MySQL or MariaDB (database management systems)
  - Perl, PHP, or Python (scripting languages)

MEAN

- I Node-världen är MEAN en vanlig stack.
  - *MongoDB* - The standard NoSQL database
  - *Express.js* - The default web applications framework
  - *Angular.js* - The JavaScript MVC framework used for web applications
  - *Node.js* - Framework used for scalable server-side and networking applications.

Express

- Vi ska kolla lite närmare på Express.
- The Express.js framework makes it very easy to develop an application which can be used to handle multiple types of requests like the GET, PUT, and POST and DELETE requests.
- Installera med `npm install express`

Express

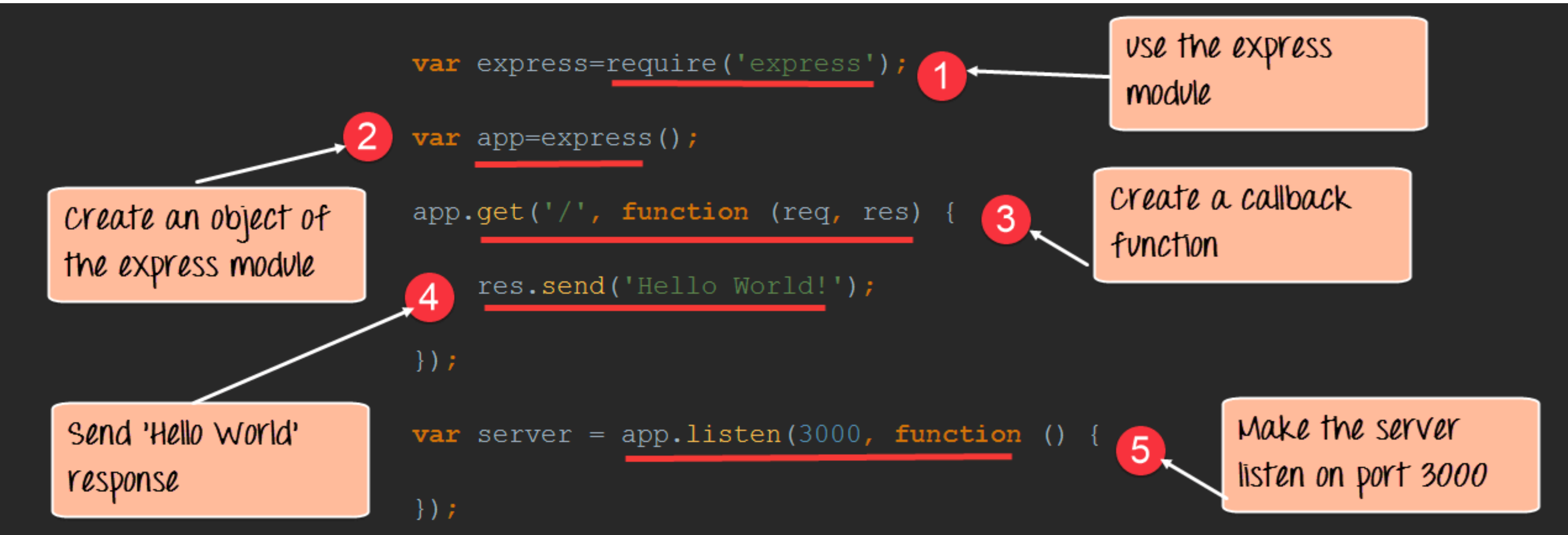
1. In our first line of code, we are using the require function to include the "express module."

2. Before we can start using the express module, we need to make an object of it.

3. Here we are creating a callback function. This function will be called whenever anybody browses to the root of our web application which is http://localhost:3000 . The callback function will be used to send the string 'Hello World' to the web page.

4. In the callback function, we are sending the string "Hello World" back to the client. The 'res' parameter is used to send content back to the web page. This 'res' parameter is something that is provided by the 'request' module to enable one to send content back to the web page.

5. We are then using the listen to function to make our server application listen to client requests on port no 3000. You can specify any available port over here.



```
var express=require('express');

var app=express();

app.get('/', function (req, res) {

    res.send('Hello World!');

});

var server = app.listen(3000, function () {

});
```

1
Use the express module
2
Create an object of the express module
3
Create a callback function
4
Send 'Hello World' response
5
Make the server listen on port 3000

Routes

- Routing determine the way in which an application responds to a client request to a particular endpoint.

- For example, a client can make a GET, POST, PUT or DELETE http request for various URL such as the ones shown below;

  ```
  http://localhost:3000/Books
  http://localhost:3000/Students
  ```

- So based on the URL which is accessed, a different functionality on the webserver will be invoked, and accordingly, the response will be sent to the client. This is the concept of routing.

- Each route can have one or more handler functions, which are executed when the route is matched.

Routes

- The general syntax for a route:

  `app.METHOD(PATH, HANDLER)`

- Wherein,

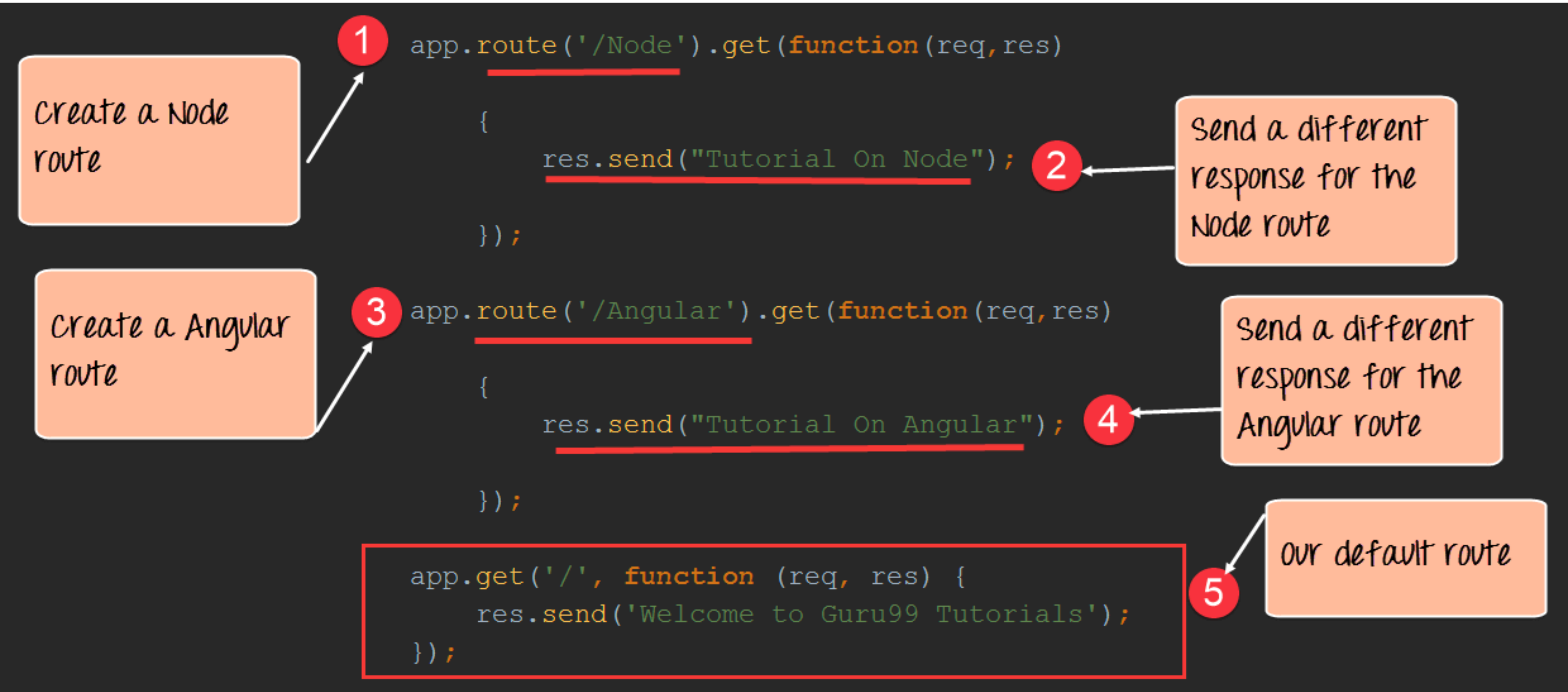
  1. *app* is an instance of the express module

  2. *METHOD* is an HTTP request method (`GET`, `POST`, `PUT` or `DELETE`)

  3. *PATH* is a path on the server.

  4. *HANDLER* is the function executed when the route is matched.

Routes


Create a Node route
1
Send a different response for the Node route
2
Create a Angular route
3
Send a different response for the Angular route
4
Our default route
5

```
app.route('/Node').get(function(req,res)

    {
        res.send("Tutorial On Node");

    });

app.route('/Angular').get(function(req,res)

    {
        res.send("Tutorial On Angular");

    });

app.get('/', function (req, res) {
    res.send('Welcome to Guru99 Tutorials');
});
```

GitHub

- Se alla dagens kodexempel på https://github.com/emmio-micke/webb19-backend1