

# 16. Sending Emails

## \* Chapter 267: How Does Sending Emails Work?





### How Does Sending Mails Work?



Udemy

- node.js and express.js are language or frameworks runtimes that we use for writing our server side logic. but with node.js you can't trivially create a mailing server. Handling mails is different to handling incoming requests and response. because it have to handle millions of mails and security and so on.
- so you will never implement your own mail server because that is a very complex task. so in reality you typically use 3rd party mail servers.
- by the way, all major web applications you might be interacting with including Udemy don't have their own mail servers. they are using 3rd party providers like AWS or whatever it's for sending emails

## \* Chapter 268: Using SendGrid

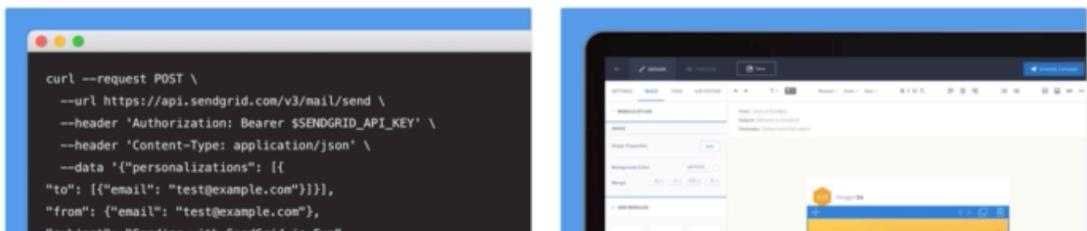




NEW Check out our tips on creating your most effective newsletter! >

## Send Shipping Notifications With Confidence

Partner with the email service trusted by developers and marketers for time-savings, scalability, and delivery expertise.

[See Plans and Pricing](#)
[Try for Free](#)


Integrate email into your app or website  
Email API

Build and send email marketing campaigns  
Marketing Campaigns

Send email with both  
Email API + Marketing Campaigns

 [Learn more](#)

### Select Your Plan

<p><input checked="" type="radio"/> <b>Free</b></p> <p>40,000 emails for your first 30 days, then send 100/day, forever</p> <p>\$0/mo</p> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Core Email API Features</li> <li><input checked="" type="checkbox"/> Core Marketing Campaigns Features</li> <li><input checked="" type="checkbox"/> Data rich email activity feed</li> <li>Dedicated IP included</li> <li>Subuser management</li> <li>Customer success manager</li> <li>Prioritized support</li> </ul>	<p><input type="radio"/> <b>Essentials</b></p> <p>Ideal for teams sending up to 100,000 emails/mo</p> <p><b>Starting at \$9.95/mo plus \$10/mo per 10,000 contacts</b></p> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Core Email API Features</li> <li><input checked="" type="checkbox"/> Core Marketing Campaigns Features</li> <li><input checked="" type="checkbox"/> Data rich email activity feed</li> <li>Dedicated IP included</li> <li>Subuser management</li> <li>Customer success manager</li> <li>Prioritized support</li> </ul>	<p><input type="radio"/> <b>Pro</b> <small>RECOMMENDED</small></p> <p>Ideal for businesses sending from 100,000 to 1.5 million emails/mo</p> <p><b>Starting at \$79.95/mo plus \$10/mo per 10,000 contacts</b></p> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Core Email API Features</li> <li><input checked="" type="checkbox"/> Core Marketing Campaigns Features</li> <li><input checked="" type="checkbox"/> Data rich email activity feed</li> <li><input checked="" type="checkbox"/> Dedicated IP included</li> <li><input checked="" type="checkbox"/> Subuser management</li> <li>Customer success manager</li> <li>Prioritized support</li> </ul>	<p><input type="radio"/> <b>Premier</b></p> <p>Ideal for businesses sending 1.5 million+ emails/mo</p> <p><b>Custom Pricing</b></p> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Core Email API Features</li> <li><input checked="" type="checkbox"/> Core Marketing Campaigns Features</li> <li><input checked="" type="checkbox"/> Data rich email activity feed</li> <li><input checked="" type="checkbox"/> Dedicated IP included</li> <li><input checked="" type="checkbox"/> Subuser management</li> <li><input checked="" type="checkbox"/> Customer success manager**</li> <li><input checked="" type="checkbox"/> Prioritized support**</li> </ul> <p><small>**for 5 million+ emails/mo</small></p>
---	--	--	--

 [See Full Plan Comparison](#)

 [Learn more](#)

- we will use it because they have a free entry tier.
  - there are many alternatives like Mailchimp, AWS, SCS etc.
- 

```

11      }
12      res.render('auth/login', {
13          path: '/login',
14          pageTitle: 'Login',
15          errorMessage: message
16      });
17  };
18
19 exports.signup = (req, res, next) => {
20     let message = req.flash('error');
21     if (message.length > 0) {
22         message = message[0];
23     } else {
24         message = null;
25     }
26     res.render('auth/signup', {
27         path: '/signup',
28         pageTitle: 'Signup',
29         errorMessage: message
30     });

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$  
Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$  
Maximilians-MBP:nodejs-complete-guide mschwarzmueller\$ npm install --save nodemailer nodemailer-sendgrid-transport  
(... rollbackFailedOptional: verb npm-session 0ba5268834c7f411

Ln 24, Col 20 Spaces: 2 UTF-8 LF JavaScript Prettier: ✓

- 'npm install —save nodemailer nodemailer-sendgrid-transport' that is a package that will help us with integrating sendgrid and conveniently use that together with nodemailer.
- 'nodemailer' is a package that makes sending emails from inside node.js.

## \* Chapter 269: Using Nodemailer To Send An Email

1. update
- ./controllers/auth.js





Unverified accounts are limited to 100 emails per day. Verify your account to send more. [VERIFY MY ACCOUNTS](#)

Maximilian Schwarz

Setup Guide / Integrate

### Integrate using our Web API or SMTP Relay

**Choose a setup method**

**RECOMMENDED**

**Web API**

The fastest, most flexible way to send email using languages like Node.js, Ruby, C#, and more.

**Choose**

**SMTP Relay**

The easiest way to send email. It only requires modifying your application's SMTP configuration.

**Choose**

Feedback

SendGrid from SendGrid

Hi there! One very quick question for you: App: Ask a Question Thanks!

Unverified accounts are limited to 100 emails per day. Verify your account to send more. [Verify My Account](#)

## API Keys

[Create API Key](#)

- Dashboard
- Marketing
- Templates
- Stats
- Activity
- Suppressions
- Settings**
  - Account Details
  - Alert Settings
  - API Keys**
  - Inbound Parse
  - IP Access Management
  - IP Addresses
  - Mail Settings
  - Experiments

**Get started creating API Keys**

API keys help protect the sensitive areas of your SendGrid account (e.g. contacts and account settings). To control and limit access of API users, you can create multiple API keys, each with different permissions.

**SendGrid from SendGrid**  
Hi there! One very quick question for you: App: Ask a Question Thanks!

- go to settings and go to API Keys and click 'Create API Key'.  
  
  


## Create API Key

API Key Name \* node-shop

API Key Permissions \* [?](#)

- Full Access**  
Allows the API key to access GET, PATCH, PUT, DELETE, and POST endpoints for all parts of your account, excluding billing.
- Restricted Access**  
Customize levels of access for all parts of your account, excluding billing.
- Billing Access**  
Allows the API key to access billing endpoints for the account. (This is especially useful for Enterprise or Partner customers looking for more advanced account management.)

[Cancel](#) [Create & View](#)

**SendGrid from SendGrid**  
Hi there! One very quick question for you: App: Ask a Question Thanks!

Unverified accounts are limited to 100 emails per day. Verify your account to send more. [Verify My Account](#)

Maximilian Schreiber

- Dashboard
- Marketing
- Templates
- Stats
- Activity
- Suppressions
- API Keys**
- Inbound Parse
- IP Access Management
- IP Addresses
- Mail Settings
- Experiments

**API Key Created**

Please copy this key and save it somewhere safe.  
For security reasons, we cannot show it to you again

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

Copied!

Done

Feedback

SendGrid from SendGrid  
Hi there! One very quick question for you: App: Ask a Question Thanks!

EXPLORER

- NODEJS-COMPLETE-GUIDE**
  - .vscode
  - controllers
    - admin.js
    - auth.js
    - error.js
    - shop.js
  - data
  - middleware
    - is-auth.js
  - models
    - order.js
    - product.js
    - user.js
  - node\_modules
  - public
  - routes
    - admin.js
    - auth.js
    - shop.js
  - util
  - views
    - admin
- OUTLINE

```

auth.js
1 const bcrypt = require('bcryptjs');
2 const nodemailer = require('nodemailer');
3 const sendgridTransport = require('nodemailer-sendgrid-transport');
4
5 const User = require('../models/user');
6
7 const transporter = nodemailer.createTransport(sendgridTransport{
8   auth: {
9     api_key: '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'
10   }
11 });
12
13 exports.getLogin = (req, res, next) => {
14   let message = req.flash('error');
15   if (message.length > 0) {
16     message = message[0];
17   } else {
18     message = null;
19   }
20   res.render('auth/login', {
21     message
22   });
23 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[nodemon] 1.18.3  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching: \*\*  
[nodemon] starting `node app.js`  
(node:87384) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version  
. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.

Ln 9, Col 84 Spaces: 2 UTF-8 LF JavaScript Prettier: ✓







- now you could use real email address. and i'm redirected.
- and now check your email account. and in that email account, you should have an email from [shop@nodecomplete.com](mailto:shop@nodecomplete.com) with that message.
- if you don't get it, verify you entered a correct e-mail.

```
1 //./controllers/auth.js
2
3 const bcrypt = require('bcryptjs');
4 const nodemailer = require('nodemailer')
5 const sendgridTransport = require('nodemailer-sendgrid-transport')
6
7 const User = require('../models/user');
8
```

```
9  /**initialize a couple of things
10 * for nodemailer, we need to initialize a so-called transporter.
11 * which is some setup telling nodemailer how your e-mails will be delivered
12 *
13 * execute 'sendgridTransport' as a function
14 * because this function will then return configuration that nodemailer can use to use
15 * sendgrid.
16 *
17 * with the transporter configured,
18 * you can now use it to send an email
19 * and i wanna send a e-mail after signing up.
20 * so go to 'postSignup'
21 */
22 const transporter = nodemailer.createTransport(sendgridTransport({
23   auth: {
24     /**both are values you get from inside your sendgrid account */
25     api_key: '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'
26   }
27 }));
28
29 exports.getLogin = (req, res, next) => {
30   let message = req.flash('error')
31   if(message.length > 0){
32     message = message[0]
33   } else {
34     message = null
35   }
36   res.render('auth/login', {
37     path: '/login',
38     pageTitle: 'Login',
39     errorMessage: message
40   });
41 }
42 exports.getSignup = (req, res, next) => {
43   let message = req.flash('error')
44   if(message.length > 0){
45     message = message[0]
46   } else {
47     message = null
48   }
49   res.render('auth/signup', {
50     path: '/signup',
51     pageTitle: 'Signup',
52     errorMessage: message
53   });
54 }
55
56 exports.postLogin = (req, res, next) => {
57   const email = req.body.email;
58   const password = req.body.password;
59   User.findOne({ email: email })
60     .then(user => {
61       if (!user) {
62         req.flash('error', 'Invalid email or password.')
63         return res.redirect('/login');
```

```

64 }
65 bcrypt
66 .compare(password, user.password)
67 .then(doMatch => {
68     if (doMatch) {
69         req.session.isLoggedIn = true;
70         req.session.user = user;
71         return req.session.save(err => {
72             console.log(err);
73             res.redirect('/');
74         });
75     }
76     req.flash('error', 'Invalid email or password.')
77     res.redirect('/login');
78 })
79 .catch(err => {
80     console.log(err);
81     res.redirect('/login');
82 });
83 }
84 .catch(err => console.log(err));
85 };
86
87 exports.postSignup = (req, res, next) => {
88     const email = req.body.email;
89     const password = req.body.password;
90     const confirmPassword = req.body.confirmPassword;
91     User.findOne({ email: email })
92     .then(userDoc => {
93         if (userDoc) {
94             req.flash('error', 'E-Mail exists already, please pick a different one. ')
95             return res.redirect('/signup');
96         }
97         return bcrypt
98             .hash(password, 12)
99             .then(hashedPassword => {
100                 const user = new User({
101                     email: email,
102                     password: hashedPassword,
103                     cart: { items: [] }
104                 });
105                 return user.save();
106             })
107             .then(result => {
108                 res.redirect('/login');
109                 return transporter.sendMail({
110                     /**you pass a javascript object
111                     * where you configure the email you wanna send
112                     */
113                     to: email,
114                     from: 'shop@nodecomplete.com',
115                     subject: 'Signup Succeeded!',
116                     html: '<h1>You Successfully Signed Up!</h1>'
117                 })
118             })
119             .catch(err => {

```

```

120     ...
121     ...
122   })
123   .catch(err => {
124     console.log(err);
125   });
126 };
127
128 exports.postLogout = (req, res, next) => {
129   req.session.destroy(err => {
130     console.log(err);
131     res.redirect('/');
132   });
133 };

```

## \* Chapter 270: Potential Limitation For Large Scale Apps



```

EXPLORER auth.js login.ejs signup.ejs app.js admin.js shop.js ...
NODEJS-COMPLETE-GUIDE
  .vscode
    controllers
      admin.js
      auth.js
      error.js
      shop.js
    data
    middleware
      is-auth.js
    models
      order.js
      product.js
      user.js
  node_modules
  public
  routes
    admin.js
    auth.js
    shop.js
  util
  views
    admin
  OUTLINE
  PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
  1: node
  Ln 105, Col 14 (209 selected) Spaces: 2 UTF-8 LF JavaScript Prettier: ✓

```

The screenshot shows the VS Code interface with the auth.js file open. The code is as follows:

```

const user = new User({
  email: email,
  password: hashedPassword,
  cart: { items: [] }
});
return user.save();
})
.then(result => {
  res.redirect('/login');
  return transporter.sendMail({
    to: email,
    from: 'shop@node-complete.com',
    subject: 'Signup succeeded!',
    html: '<h1>You successfully signed up!</h1>'
  });
}
.catch(err => {
  console.log(err);
});
}

```

A tooltip is shown over the `transporter.sendMail()` call, highlighting it. The status bar at the bottom indicates "Ln 105, Col 14 (209 selected) Spaces: 2 UTF-8 LF JavaScript Prettier: ✓".

- it's good that we don't block the redirect but that we direct and send the mail at the same time because if you have an application with a lot of requests and you would wait for the email to be sent before you redirect, you might slow down your application because you are sending a lot of emails.
- depending on the size of your app we are talking about really huge apps, you could look into totally different approaches where you have some server side scripts running every x hours or every x minutes that send e-mails to newly signed users.
- so you should consider not using this in a blocking way because if you wait for this to be finished, this can be slow and you have to evaluate if it's worth waiting this or if your user can continue even without that mail being delivered.