Popular Express Middleware

- body-parser: parse HTTP request body
- compression: compress HTTP responses
- cookie-parser: parse cookie header and populate req.cookies
- multer: handle multipart form data
- serve-favicon: serve a favicon
- session: establish server-based sessions (development only)
- helmet: helps secure your apps by setting various HTTP headers
- passport: authentication using "strategies" such as OAuth,
 OpenID, and many others

Using bcrypt to Hash and Compare Passwords

https://auth0.com/blog/hashing-in-action-understanding-bcrypt



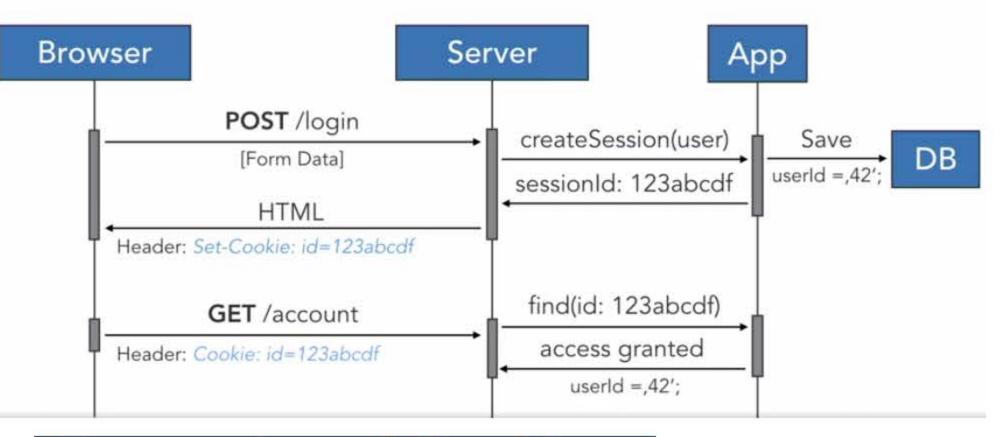
Version and Rounds Salt Hash



Understanding

Cookies and Sessions

Authentication Flow



Properties of Cookies

Set-Cookie:

- id=123abcdf;
- Expires=Wed, 14 Nov 2018 00:30:00 GMT;
- Secure;
- HttpOnly;

Reading Cookies in Express

```
const cookieParser = require(,cookie-parser');
app.use(cookieParser()); // gives us req.cookies
```

Handling Sessions in Express

```
const session = require(,express-session');
const MongoStore = require('connect-mongo')(session);
app.use(session({
   secret: 'foo',
   store: new MongoStore(options)
}));
```

Introduction to Passport

Conceptual Overview



Using Passport in Express

passport.initialize();

Returns a middleware function that uses the request (req) object to store passport internal data in it.

passport.session();

Looks for a previously serialized user in the current session and uses a provided descrialization function to provide the user in req.user to all following middlewares and routes.

Authentication vs. Authorization

Authentication

Who is this user?

Authorization

What is the user allowed to do?

Protecting Routes

Tuning Express Performance

Set NODE_ENV to production





NODE_ENV=development

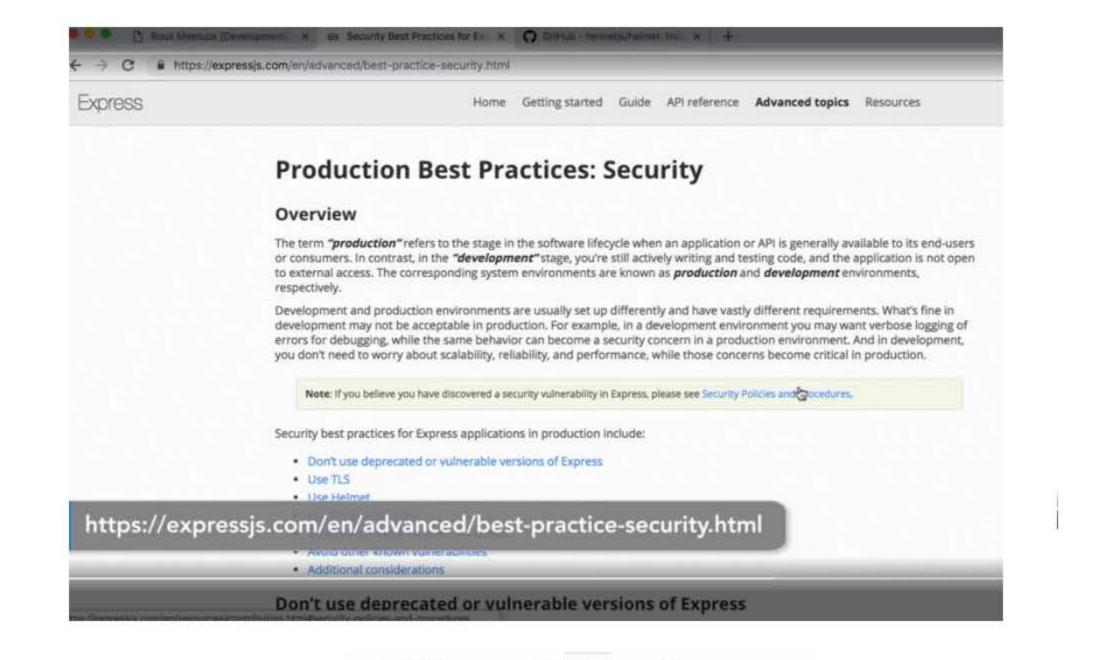
NODE_ENV=production

The drastic effects of omitting NODE_ENV in your Express.js applications Daniel Khan in Digital experience July 22, 2015 Most developers learn best by examples, which naturally tend to simplify matters and omit things that aren't essential for understanding. This means that the "Hello World" example, when used as starting point for an application, may be not suitable for production scenarios at tarted using Node is like that and I have to confess that it took me almost two years to quantify the huge performance impact of omitting a single environment variable. In fact it was

Compress Server Responses

```
// npm install —save compression
const compression = require('compression');
const express = require('express');
const app = express()
app.use(compression())
```

- Don't use synchronous functions
- Don't use console.log() because it's synchronous
- Handle errors and exceptions properly
- Use a cluster

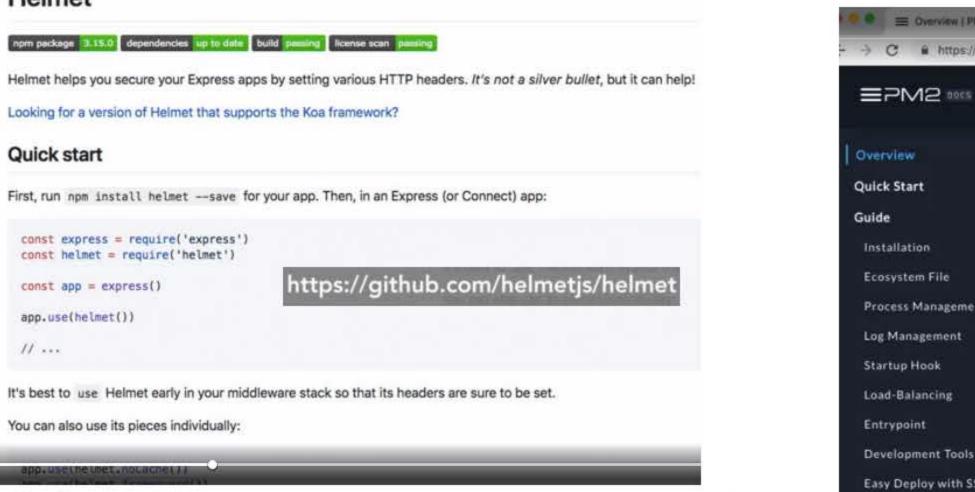


car-booth-20-mac:meetup danielkhan\$ npm audit

=== npm audit security report ===

Further Recommendations

- Cache request results



How it works

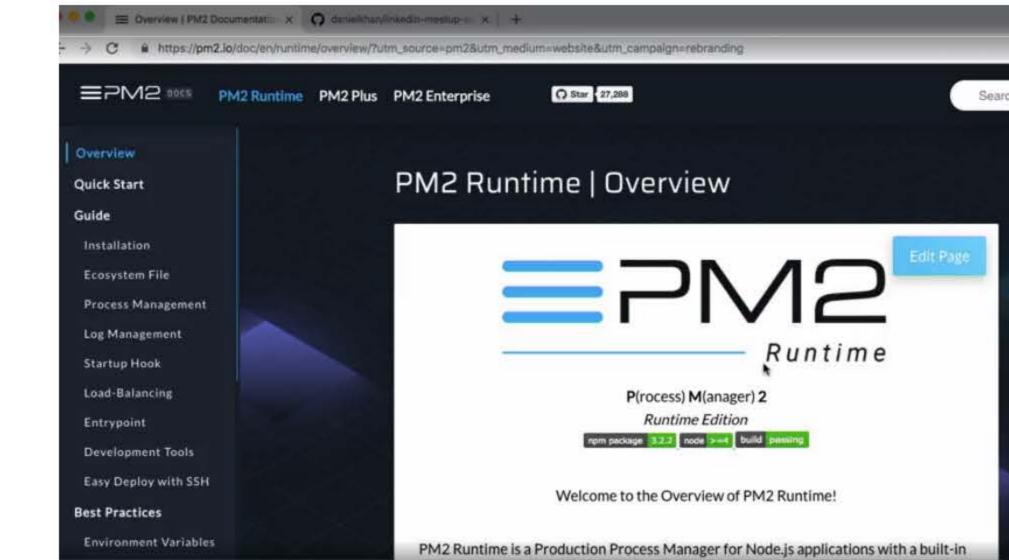
Helmet is a collection of 14 smaller middleware functions that set HTTP response headers. Running app.use(helmet()) will not include all of these middleware functions by default.

Module	Default?
contentSecurityPolicy for setting Content Security Policy	
crossdomain for handling Adobe products' crossdomain requests	
dnsPrefetchControl controls browser DNS prefetching	J
expectCt for handling Certificate Transparency	
featurePolicy to limit your site's features	
frameguard to prevent clickjacking	~
hidePoweredBy to remove the X-Powered-By header	~
hpkp for HTTP Public Key Pinning	
hsts for HTTP Strict Transport Security	~
ioNoOpen sets X-Download-Options for IE8+	V.



A hacker can use this information to their advantage. If they know of a vulnerability in Express or Node

is powered by Express.



PM2 Rely on github to push application to a server

Load Balancer. It allows you to keep applications alive forever, to reload them without

```
deploy: {
  production: {
   user: 'nodejs',
   host: 'meetup.demo.khan.io',
   ref: 'origin/master'
   repo: 'https://github.com/danielkhan/linkedin-meetup-sample',
   path: '/home/nodejs/deploy',
    'post-deploy': 'cp ../.env ./ && npm install && pm2 startOrRestart ecosystem.con
```

```
drwxr-xr-x 4 nodejs root 4.8K Dec 6 22:19 .
drwxr-xr-x 11 nodejs nodejs 4.8K Dec 6 21:87 .
-rw-rw-r- 1 nodejs nodejs 8 Dec 6 22:19 .deploys
-rw-rw-r- 1 nodejs nodejs 323 Dec 6 21:29 .env
lrwxrwxrwx 1 nodejs nodejs 26 Dec 6 22:19 current -> /home/nodejs/deploy/sou
 drwxrwxr-x 4 nodejs nodejs 4.8K Dec 6 22:18 shared drwxrwxr-x 9 nodejs nodejs 4.8K Dec 6 22:19 source
  nodejs@ip-172-31-48-252:-/deployS pm2 status
   Name Id mode statue U cpu memory
 meetup 0 1.0.0 fork unline 3 8% 67.2 MB
   PM2] Saving current process list...
  [PM2] Successfully saved in /home/nodejs/.pm2/dump.pm2
 modeis@ip-172-31-48-252:-/deploy$ pm2 startup
 [PH2] Init System found: systemd
[PH2] To setup the Startup Script, copy/paste the following command:
sudo env PATH=SPATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -
  u nodejs --hp /home/nodejs
  nodejs@ip-172-31-48-252:-/deployS sudo env PATH=SPATH:/usr/bin /usr/lib/node_mod
 ules/pm2/bin/pm2 startup systemd -u nodejs --hp /home/nodejs
[sudo] password for nodejs:
  ExecReload=/usr/lib/node_modules/pm2/bin/pm2 reload all
  ExecStop=/usr/lib/node_modules/pm2/bin/pm2 kill
 WantedBy=multi-user.target
  Target path
   /etc/systemd/system/pm2-nodejs.service
  [PM2] Writing init configuration in /etc/systemd/system/pm2-nodejs.service
   [PM2] Making script booting at startup...
 [PM2] [-] Executing: systemctl enable pm2-nodejs...
Created symlink /etc/systemd/system/multi-user.target.wants/pm2-nodejs.service +
  /etc/systemd/system/pm2-nodejs.service.
   [PM2] [v] Command successfully executed.
  [PM2] Freeze a process list on reboot via:
 [DH2] Remove init script via:
 $ pm2 unstartup systemd
nodejs@ip-172-31-48-252:~/deploy$ |
```

odejs@ip-172-31-48-252:-/deploy\$ pwd

Best practice for Prod: run node application behing web proxy such as NGINX - better to manage SSL and TLS Prod Web Server: NGINX

```
Provide free ssl self-signed certificate to
server
```

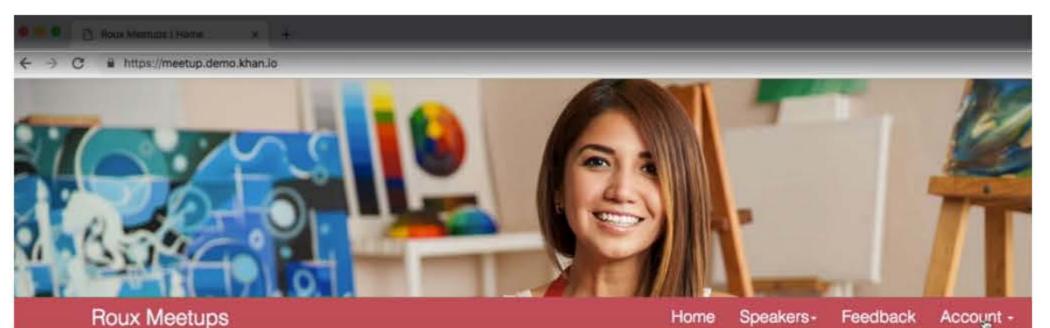
Proxy configuration on, nginx

server_name meetup.demo.khan.io; # managed by Certbot

location / {
 proxy_redirect off;

```
proxy_http_version 1.1;
proxy_pass http://127.8.8.1:3888;
             proxy_set_header Host Shost;
            proxy_set_header X-Real-IP Sremote_addr;
proxy_set_header X-Forwarded-For Sproxy_add_x_forwarded_for;
              proxy_set_header X-Forwarded-Proto Sscheme;
       listen [::]:443 ssl ipv6only=on; # managed by Certbot
listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/meetup.demo.khan.io/fullchain.pem; # managed by Certbot
         include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
       ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
   server {
   if ($host = meetup.demo.khan.io) {
           return 301 https://Shost$request_uri;
          listen 80 ;
listen [::]:80 ;
       server_name meetup.demo.khan.io;
       return 484; # managed by Certbo
if (app.get('env') === 'production') {
  app.set('trust proxy', 'loopback');
   app.use(session({
    secret: 'another very secret 12345',
    name: 'sessionId',
    proxy: true,
     cookie: { secure: true },
     resave: true,
     saveUninitialized: false,
    store: new MongoStore({ mongooseConnection: mongoose.connection }),
} else {
```

car-booth-20-mac:meetup danielkhan\$ pm2 update car-booth-20-mac:meetup danielkhan\$ pm2 deploy production



Advanced Express

with Daniel Khan



Setting Up a Deployment Environment

- 1. Linux host (example: an EC2 instance with Ubuntu)
- 2. Install Node.js and PM2 (npm install -g pm2).
- 3. /home/nodejs/deploy owned by a user "nodejs"
- 4. Add local SSH public key to authorized_keys on server.
- 5. Push your project to a Git repository (example: GitHub).
- 6. Copy .env to the deployment directory.
- 7. Whitelist the server IP on MongoDB Atlas.