(https://telegram.me/evdokimovm) (https://twitter.com/id194695811) (https://github.com/evdokimovm)


evdokimovm.github.io (/)

Create Pagination with Node.js, MongoDB, Express and EJS Step by Step from Scratch

AUGUST 20TH, 2017

Suppose we have a collection contains many of data (products for instance) and we need to view all on website. We can just output products on page but this looks not so clean. In such cases we need a pagination.

In this article I'll tell about how to make pagination using Node.js, MongoDB, Express.js, EJS and Bootstrap from scratch. We will create new database, declare data collection, fill this collection with data and output contents to page with pagination.

This article is generally for beginners who interested in web development so I will try to make all explanations very detailed. But maybe this article will be useful for an experienced reader.

After reading this article, you will be able to create pagination from scratch, **not like this**:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110
111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156
157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202
203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225
226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248
249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271
272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294
295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317

it's a **bad practice**.

You will be able to create pagination from scratch like this:

First ... 146 147 148 149 150 151 152 153 154 ... Last

For reading this article you need to have installed Node.js and MongoDB. You can download and install it from official websites:

Node.js - https://nodejs.org/en/ (https://nodejs.org/en/)

MongoDB - https://www.mongodb.com/download-center?jmp=nav#community (https://www.mongodb.com/download-center?jmp=nav#community)

Get started by creating a new folder for this project, and name it anything you like. Then, inside that folder, create additional folders and files to match the following structure:

```
.
├── routes
|    └── main.js
├── models
|    └── product.js
├── views
|    ├── main
|    |    ├── products.ejs
|    |    └── add-product.ejs
|    └── index.ejs
└── server.js
```

and install necessary dependencies to work:

```
npm install --save mongoose express body-parser ejs faker
```

Create a Server and Connect to MongoDB

Open `server.js` file and create simple express.js server:

```
var express = require('express')
var ejs = require('ejs')
var mongoose = require('mongoose')
var bodyParser = require('body-parser')

var app = express()

mongoose.connect('mongodb://localhost:27017/article')

app.use(bodyParser.json())
app.use(bodyParser.urlencoded({
    extended: true
}))

app.set('view engine', 'ejs')

app.listen(8080, function() {
    console.log('Node.js listening on port ' + 8080)
})
```

In this file we create connection to MongoDB, enable `body-parser` middleware for parse incoming request bodies and use `ejs` template engine.

Declare Collection and Fill DataBase

Before create routes to filling our database we need declare model for products collection. Products will have a name, category name, price and cover image.

Open `models/product.js` and paste following code:

```
var mongoose = require('mongoose')
var Schema = mongoose.Schema

var ProductSchema = new Schema({
    category: String,
    name: String,
    price: Number,
    cover: String
})

module.exports = mongoose.model('Product', ProductSchema)
```

> Later In this article I will provide the code for quickly filling the database with fake data using faker.js

Now we can create a `POST` route and new page with HTML form to fill collection with products.

Open `routes/main.js` file and paste this code:

```
var router = require('express').Router()
var faker = require('faker')
var Product = require('../models/product')

router.get('/add-product', function(req, res, next) {
    res.render('main/add-product')
})

router.post('/add-product', function(req, res, next) {
    var product = new Product()

    product.category = req.body.category_name
    product.name = req.body.product_name
    product.price = req.body.product_price
    product.cover = faker.image.image()

    product.save(function(err) {
        if (err) throw err
        res.redirect('/add-product')
    })
})

module.exports = router
```

In this code we created `GET` route to show *add-products* page with HTML form and `POST` route that we will use to add new products to MongoDB. Also in this case as product cover image I use fake image from faker.js library.

> About how to upload files using Node.js and Multer you can read in my article: Upload files to server using Node.js and Multer package (https://evdokimovm.github.io/javascript/nodejs/expressjs/multer/2016/11/03/Upload-files-to-server-using-NodeJS-and-Multer-package-filter-upload-files-by-extension.html).

We need to use this routes in our server. To do that add following code to `server.js` file:

```
var mainRoutes = require('./routes/main')
app.use(mainRoutes)
```

Let's create `add-product` page with form to send a `POST` request to our `POST` route.

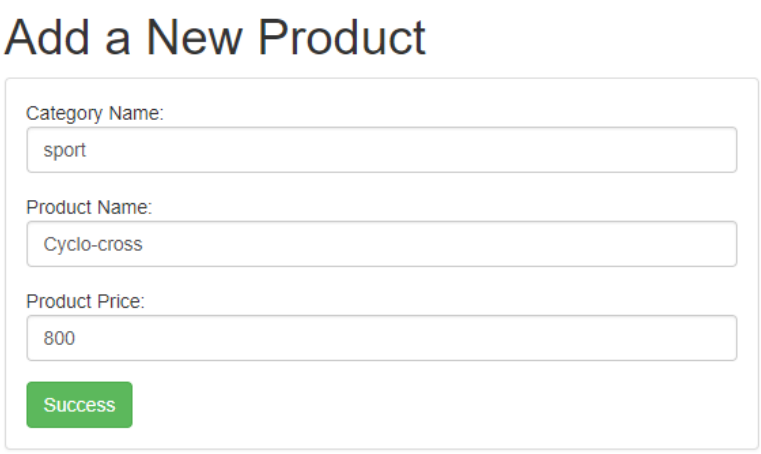
Open `views/main/add-product.ejs` file and paste following HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title></title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstra
p.min.css">
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-lg-6 col-md-offset-3">
                <h1>Add a New Product</h1>
                <form method="post">
                    <div class="panel panel-default">
                        <div class="panel-body">
                            <div class="form-group">
                                <label for="category_name">Category Name: </label>
                                <input type="text" id="category_name" class="form-control" nam
e="category_name">
                            </div>
                            <div class="form-group">
                                <label for="product_name">Product Name: </label>
                                <input type="text" id="product_name" class="form-control" name
="product_name">
                            </div>
                            <div class="form-group">
                                <label for="product_price">Product Price: </label>
                                <input type="text" id="product_price" class="form-control" nam
e="product_price">
                            </div>
                            <button type="submit" class="btn btn-success">Success</button>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</body>
</html>
```

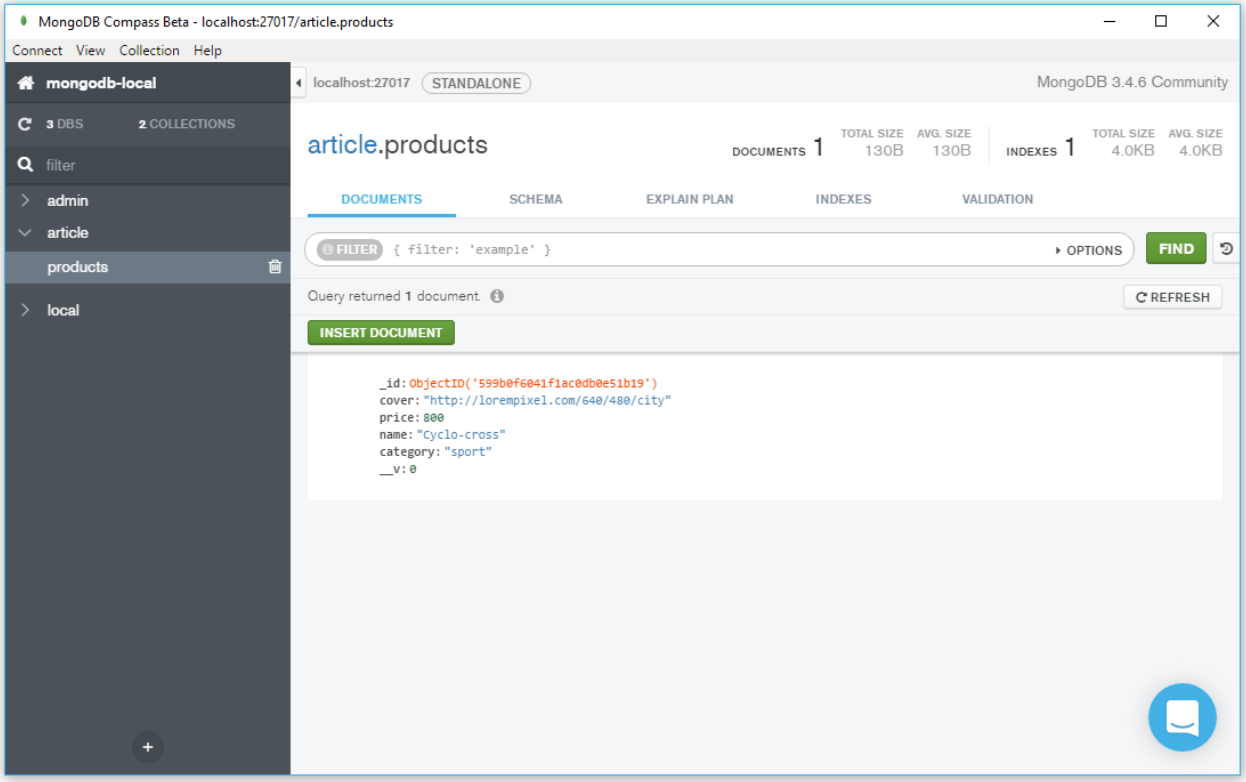
Now you can run your MongoDB, then run server with `node server.js` command and go to `http://localhost:8080/add-product` page in your browser and add new items to MongoDB.

For convenient viewing and editing MongoDB contents on your desktop I recommend to use MongoDB Compass (https://www.mongodb.com/download-center?jmp=nav#compass).

Create new product:



See in MongoDB Compass:



Quick Filling DataBase Using Faker.js

Open `routes/main.js` file and paste following code:

```
router.get('/generate-fake-data', function(req, res, next) {
    for (var i = 0; i < 90; i++) {
        var product = new Product()

        product.category = faker.commerce.department()
        product.name = faker.commerce.productName()
        product.price = faker.commerce.price()
        product.cover = faker.image.image()

        product.save(function(err) {
            if (err) throw err
        })
    }
    res.redirect('/add-product')
})
```

Each time you will go to `http://localhost:8080/generate-fake-data` to your database will insert 90 new items. After filling the database, you will be redirected to `add-product` page.

Output Products to Page with Pagination

First we need to create new route to render `products` page and output some necessary data to create pagination. Open `routes/main.js` file and paste following code:

```
router.get('/products/:page', function(req, res, next) {
    var perPage = 9
    var page = req.params.page || 1

    Product
        .find({})
        .skip((perPage * page) - perPage)
        .limit(perPage)
        .exec(function(err, products) {
            Product.count().exec(function(err, count) {
                if (err) return next(err)
                res.render('main/products', {
                    products: products,
                    current: page,
                    pages: Math.ceil(count / perPage)
                })
            })
        })
})
```

Explanation:

`perPage` variable contains max number of items on each page, `page` variable contains current page number.

First we are finding all documents in `Products` collection:

```
Product
    .find({})
```

for each page we need to skip `((perPage * page) - perPage)` values (on the first page the value of the skip should be 0):

```
.skip((perPage * page) - perPage)
```

output only `perPage` items (9 in this case):

```
.limit(perPage)
```

count all items in collection with `count()` (we will use this value to calculate the number of pages):

```
Product.count().exec(function(err, count) {
    ...
```

then render `main/products` page and send necessary data:

```
res.render('main/products', {
    products: products,
    current: page,
    pages: Math.ceil(count / perPage)
})
```

Create pagination page:

Open `views/main/products.ejs` and paste following EJS:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
    <link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.7/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <div class="row">
            <% for (var i = 0; i < products.length; i++) { %>
                <div class="col-md-4">
                    <div class="thumbnail">
                        <img src="<%= products[i].cover %>">
                        <div class="caption">
                            <h3><%= products[i].name %></h3>
                            <p><%= products[i].category %></p>
                            <p>$<%= products[i].price %></p>
                        </div>
                    </div>
                </div>
            <% } %>
        </div>
            <% if (pages > 0) { %>
                <ul class="pagination text-center">
                    <% if (current == 1) { %>
                        <li class="disabled"><a>First</a></li>
                    <% } else { %>
                        <li><a href="/products/1">First</a></li>
                    <% } %>
                    <% var i = (Number(current) > 5 ? Number(current) - 4 : 1) %>
                    <% if (i !== 1) { %>
                        <li class="disabled"><a>...</a></li>
                    <% } %>
                    <% for (; i <= (Number(current) + 4) && i <= pages; i++) { %>
                        <% if (i == current) { %>
                            <li class="active"><a><%= i %></a></li>
                        <% } else { %>
                            <li><a href="/products/<%= i %>"><%= i %></a></li>
                        <% } %>
                        <% if (i == Number(current) + 4 && i < pages) { %>
                            <li class="disabled"><a>...</a></li>
                        <% } %>
                    <% } %>
                    <% if (current == pages) { %>
                        <li class="disabled"><a>Last</a></li>
                    <% } else { %>
                        <li><a href="/products/<%= pages %>">Last</a></li>
                    <% } %>
                </ul>
            <% } %>
    </div>
</body>
</html>
```

Explanation:

We will show pagination only if number of pages more than 0:

```
<% if (pages > 0) { %>
```

In this part of code:

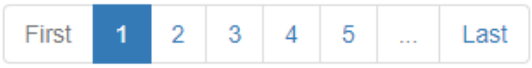
```
<% var i = (Number(current) > 5 ? Number(current) - 4 : 1) %>
```

we are check number of current page and if this value less than 5 we are output pagination links from 1 to `current_page + 4`:

```
<% for (; i <= (Number(current) + 4) && i <= pages; i++) { %>
```

and if this value more than 5 we will start output pagination link from `current_page - 4` to `current_page + 4`, thus previous links on `current_step - 5` will be hide.

Gif Demo:



Additional condition `&& i <= pages` in `for` loop we are using for not to display the number of links more than the number of pages.

Here:

```
<% if (i !== 1) { %>
    <li class="disabled"><a>...</a></li>
<% } %>
```

we check the number from which we begin the output and if this number not qual to `1` (read above) we are prepend `...` (to show that we have previous links).

The same way we are using at the end:

```
<% if (i == Number(current) + 4 && i < pages) { %>
    <li class="disabled"><a>...</a></li>
<% } %>
```

to append `...` to show that we have links ahead.

Finally at start and the end of pagination we toggle active of `Fist` and `Last` links:

```
<% if (current == 1) { %>
    <li class="disabled"><a>First</a></li>
<% } else { %>
    <li><a href="/page/1">First</a></li>
<% } %>

...

<% if (current == pages) { %>
    <li class="disabled"><a>Last</a></li>
<% } else { %>
    <li><a href="/page/<%= pages %>">Last</a></li>
<% } %>
```

At the end let's add some contents to `views/index.ejs` file:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
    <link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.
3.7/css/bootstrap.min.css">
</head>
<body>
    <div class="container" style="margin-top: 50px">
        <div class="row">
            <div class="col-lg-12 col-md-offset-3">
                <div class="col-md-4">
                    <a href="/add-product" class="btn btn-primary btn-lg">Add Product</a>
                </div>
                <div class="col-md-4">
                    <a href="/products/1" class="btn btn-primary btn-lg">Pagination Page</a>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

and add new route to render this page to `routes/main.js` file:

```
router.get('/', function(req, res, next) {
    res.render('index')
})
```

Now you can run MongoDB with `mongod` command, `node server.js` command, go to `http://localhost:8080/generate-fake-data` to insert data to MongoDB and use pagination on `http://localhost:8080/products/1` page.

JavaScript (14) , (/categories.html#JavaScript-ref) NodeJS (10) , (/categories.html#NodeJS-ref) MongoDB (2) , (/categories.html#MongoDB-ref) Pagination (1) , (/categories.html#Pagination-ref) ExpressJS (4) , (/categories.html#ExpressJS-ref) EJS (1) , (/categories.html#EJS-ref) Bootstrap (1) (/categories.html#Bootstrap-ref)

JavaScript (14) , (/tags.html#JavaScript-ref) NodeJS (10) , (/tags.html#NodeJS-ref) MongoDB (2) , (/tags.html#MongoDB-ref) Pagination (1) , (/tags.html#Pagination-ref) ExpressJS (4) , (/tags.html#ExpressJS-ref) EJS (1) , (/tags.html#EJS-ref) Bootstrap (1) (/tags.html#Bootstrap-ref)

Share Post

Twitter (https://twitter.com/share?text=Create Pagination with Node.js, MongoDB, Express and EJS Step by Step from Scratch&via=id194695811)

Facebook (https://www.facebook.com/sharer/sharer.php) Google+

Mikhail

I am Mikhail Evdokimov, a Web Developer and Perfectionist from Russia, Far East, Vladivostok. Enthusiast performance, responsive design and usability, always looking for the best project and good coffee. Love JavaScript and JavaScript Full Stack Development. MongoDB, ExpressJS,

AngularJS, NodeJS. Also I can use a few Ruby and Lua for Corona SDK. In my blog, I plan to write not only about development but also about my other hobbies and interests.

← Previous (/windows/zsh/shell/syntax/highlighting/ohmyzsh/hyper/terminal/2017/02/24/how-to-install-zsh-and-oh-my-zsh-on-windows-10.html)

Next →

The Disqus comment system is temporarily in maintenance mode. You can still read comments during this time, however posting comments and other actions are temporarily delayed. ×

34 Comments evdokimovm 1 Login

Recommend 8 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

Trafalgar D. Water Law • a year ago
Best article I could find about pagination. Thanks a lot! needed this for my web app.
3 • Reply • Share ›

evdokimovm ➚ Trafalgar D. Water Law • a year ago
Glad it Helped
• Reply • Share ›

CC • 7 months ago
GitHub link of this amazing article is available ?
1 • Reply • Share ›

Danh Nguyen • 20 days ago
Simply amazing no-BS tutorial! Thank you so much! https://repl.it/@NguyenDa18...
• Reply • Share ›

Rica y Feliz • a month ago
Thank you so much, I am Angela from Perú, I am doing it using dust.js for the first and the last page I can do but for intermiates I couldn't, so I would like to know if you can Help me, this is my template dust.js:

{@gt key=pages value="0" type="number"}

```
<ul class="pagination-box">
{@eq key=current value="1" type="number"}
<li class="disabled">Previo
</li>
{:else}
<li class="active">1</li>
{/eq}

<li>2</li>
<li>3</li>

{@eq key=current value="{pages}" type="number"}
<li class="disabled">
Siguiente
```

see more

Reply • Share ›

Tope shittu • 2 months ago

Thanks a lot. This is pure awesome

Reply • Share ›

Anthony Oyathelemhi • 6 months ago

This was a really good tutorial. Thanks for this

Reply • Share ›

Aarish Rahman • 6 months ago

Can you provide an example of how to do pagination using mongoose, express and express-handlebars

Reply • Share ›

Carl J Summers • 7 months ago

I wish I knew about this tutorial before I got started on my portfolio app. Pagination seems to come up often and I know I can do this for next time. Thanks for the tutorial!

Reply • Share ›

Hoang Toi • 7 months ago

Great. Thanks

Reply • Share ›

Mher Margaryan • 9 months ago

Спасибо большое, выручили.

Reply • Share ›

Richard Messi • 9 months ago

Hi Great Article...
I understood except for the part where "Number" is being used in the line " <% var i = (Number(current) > 5 ? Number(current) - 4 : 1) %> "

Where does this "Number" come from??

Sorry for the newbie question, I am learning pagination for the first time

Reply • Share ›

evdokimovm ➚ Richard Messi • 9 months ago

Number() used here for converting a string to a number. Because number of "current" page we have as string from server

https://developer.mozilla.o...

Reply • Share ›

Sam Alam • 10 months ago

Nice Article, looking for what i've expected. Cheers Author, really helpful

Reply • Share ›

Matias • 10 months ago

what if the page number is part of the query string? like mypage.com/product?color=re...
how would you recreate all the infinite parameter than a "product" can receive as a parameter and then update only the "page" param?

Reply • Share ›

Nolan James Harris • 10 months ago

What a great article!! Thanks for "dumbing it down" a bit for us newbs! I really appreciate the info. After researching and googling pagination with mongo and express I was feeling like it MUST be harder than i originally expected, but you proved that it didnt have to be! Thanks again!

Reply • Share ›

Charan KA • 10 months ago

Thanks, you made my Day!

Reply • Share ›

Hadro • 10 months ago

Great article! I have it working but I want to reverse the order to show the most recent items added to mongo first. I created a variable and saved the arr.reverse() to it but when I replace Product.find({}) with that variable my app crashes. Any ideas?

Reply • Share ›

Deva • a year ago

really good

Reply • Share ›

Mr~Kdrama • a year ago

I wonder how you would do that when you get the page information from the http header through an API call? :o like a list of repositories throught github api

Reply • Share ›

Sarthak Agarwal • a year ago

Thanks a lot sir! It is really helpful.

Reply • Share ›

Neto Deolino • a year ago
Great... Thank you!!!
Reply • Share ›

STRONK_PL • a year ago
Thanks dude, I'm working with mysql but could make it work, awesome tutorial!
Reply • Share ›

Muralidharan Chinnasamy ➚ STRONK_PL • 7 days ago
how man!! please explain it,...
Reply • Share ›

Rajesh Barik • a year ago
Loved this article... Really helped me... Thank you...
Reply • Share ›

sham • a year ago
Previous and next page links are not working??
Reply • Share ›

evdokimovm ➚ sham • a year ago
I don't wrote about "prev" and "next" links
Reply • Share ›

sham ➚ evdokimovm • a year ago
So how can we include prev and next button??
Reply • Share ›

evdokimovm ➚ sham • a year ago
https://pastebin.com/utSkjmBf

Preview button added on lines: 31 - 33
Next button added on lines: 48 - 50

This article is helped for you?
1 Reply • Share ›

pradip dhakal ➚ evdokimovm • a year ago
Thank you man , thanks a lot.
This is what i need.
1 Reply • Share ›

sham ➚ evdokimovm • a year ago
thanks for the update let me try it
Reply • Share ›

evdokimovm ➚ sham • a year ago
I will update article with adding "preview" and "next" buttons later.
Reply • Share ›

Jonas • a year ago

Hi, it seems you missed too parse variable page as integer

var page = (parseInt(req.query.page)) || 1;

then it works in my project fine. Thanks for your blog post help me a lot.

Best regards
Jonas

Reply • Share ›

evdokimovm ➚ Jonas • a year ago

Glad it helped. Thanks for reading.

Reply • Share ›

ALSO ON **EVDOKIMOVM**

NodeJS process.argv command line arguments

1 comment • 2 years ago

ohoroyoi — you save my time! thank you!

How to Install Zsh and Oh My Zsh on Windows 10

73 comments • 2 years ago

Juan Jose Cano Duque — Eres un genio ! love you

© 2017 Mikhail with Jekyll. Theme: dbyll (https://github.com/dbtek/dbyll) by dbtek.