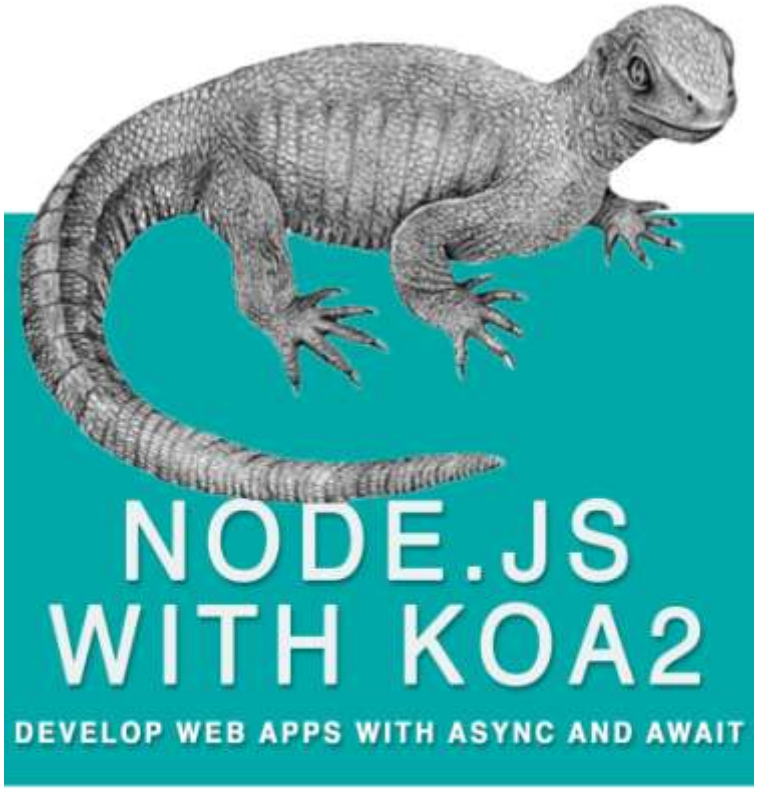

STEP BY STEP GUIDE
TO DEVELOP WEB APPS
WITH COMPLETE SOURCE CODE

Introduction

I assume you know JavaScript, knows web development, knows about node.js, tried it, have idea about Angularjs, Reactjs.

If you are trying to learn node.js, this book will help.

I will make your nodejs learning very easy, after reading this book, you will be able to make entire websites with nodejs using koa framework.

When I tried to learn node with koa, not much help is available online, there were absolutely not enough tutorials to help you learn, even to this date when iam writing this book, hardly any help will be there to help you build entire website from scratch, so iam writing this eBook, to help you learn faster.

Node with Koa is very easy, just like Meteorjs, if you know Meteorjs, you will know how coding is breeze, in Meteorjs, but then why not code in Meteorjs instead of node.

Although coding is breeze in Meteorjs, it is only for advance Linux user, hosting is pain, and you need to have at least intermediate levels of Linux skills to host it on VPS or dedicated server, there is no shared hosting service for meteor at the time of writing this eBook.

Meteor looked so promising and so easy, that I coded entire dating website in Meteorjs, and also hosted it, even though my hosting company gave me full support, did all the heavy lifting, still frequently website used to crash, and I was not sure what was the reason.

It does require Linux admin skills, Nginx skills and Docker skills.

If you can make websites on nodejs, they can be hosted on windows platform, websites are cool in nodejs, they are fast and real-time.

Future is micro services, software as a service; you will be developing software as a service not like in the past where you develop entire monolithic application with all the features in it, you will be building small service apps, and will be integrating other apps with it made by you or some other person.

Money is in micro service, you can develop a micro service app and upload it on cloud, and people will use your service and pay you as per use. For example you can develop an algorithm which can detect diabetes provided with some specific parameters, and people will pay you for using your algorithm, lot of services like this are available on cloud.

Why should you learn nodejs with koa

Nodejs is cool, websites are fast and real-time, even though you can create websites with nodejs and express, but coding is little painful, so many call backs, it's like call back jungle, even though express framework now has generator function, but it is good to use framework which is futuristic, koa is futuristic.

Future is micro services, and they are extremely easy to code in nodejs, bulk of examples and sample code will be available of nodejs, micro services are mostly event driven, nodejs is also event driven.

Nodejs and python are the important languages to learn, lot of work is going on these two languages, you will get lot of source code for free, and you just need to know how to use it.

Iam asp.net mvc C# developer, but I learned python and nodejs, since I am also learning machine learning, lot of example code for machine learning is in python and nodejs.

I also learned math's to be able to code for machine learning and artificial intelligence, artificial intelligence will be the future, so be prepare for it.

Microsoft is lagging way behind; open source community is thriving right now.

Why it was hard to learn nodejs

We came with background with class oriented programming, from C#, Java, C, c++.

JavaScript is functional programming, and we never took JavaScript seriously, we just used it for website validation.

Why you had hard time learning nodejs, nodejs is rapidly evolving technology, framework changes happens almost every day, the code you might have got searching Google is old and absolute and no longer works, always check the date when that article was written, it was working then, but not working now, you were typing the same example code, but when you run it, it doesn't work, it is most likely that example code is absolute.

Some examples on the internet are using nodejs with Angularjs or Reactjs, that is making it more problematic to learn, you got to learn additional Angularjs framework and Reactjs framework and that confuses you, where to start what to do, it totally confuses you.

There is no need to learn Angularjs, Reactjs, for building apps with nodejs, nodejs with koa framework has everything required to make a full blown web app.

What these Angularjs, Reactjs frameworks does, they add extra layer of complexity on the app, and you waste your time to figure it out, what happened, from where this error is coming, this is where I wasted my time, when I was learning.

Angularjs, and Reactjs framework, updates and change frequently, so when you try to implement there code, it doesn't work, further increasing your frustration.

There tutorial nodejs with Angularjs are written by geeks who have no life, but to experiment and do some outstanding things in coding, and we refer these tutorials for basic learning and gets stomped.

Books available on the internet are old, outdated technology, check when it was updated, nodejs is changing very rapidly, if you buy these books with outdated techniques, it will add to your frustration.

Lot of Modules are interdependent on each other, if some modules get upgraded and you update it, it may make your code error prone, since the modules which were dependent on that module are not updated or changed completely, this is the real cause of frustration.

Each module is independently developed by geeks and they are dependent on other modules to function.

When I tried to use koa with generator function, I faced lot of problems, because some modules were abandoned by their creators and were no longer working, so I used KOA2 which has functionality like async and await much like C # code; it is implementing ECMASCRIPT 6 features.

I will also explain problems I faced and reasons behind and what I did to fix it.

Note: If you are totally new then you have to look for , what packages and package versions you are using in project, don't get confuse in packages, there are so many packages to use,

Example

For Database Connection:

co-monk easy, but mongoose is also good for creating schema and predefine functions For Mongodb database connection there is Monk, Co-Monk And mongoose:

I used monk

For parsing co-body and koa-bodyparser:

I found koa-bodyparser easy you have to call just app.user(bodyparser); for parsing data from pages and using ctx.request.body u will get data from forms, but in co-body you have to mention parse(this.body) for each route

For rendering koa-swig, koa-view, co-views:

I found koa-views better, just app.user(Views('foldername',{map:{html:swig}}));

And you can render pages by calling this.render('pagename');

For routing koa-route, koa-router:

Found koa-route better but it has some limitation but I didn't found those yet

For templating, too many options are available swig, handlebars, underscore and many more

Found swig better and easy for me because you don't have to do extra coding for it {%block%} is good like handlebars you can call {{datafromRoute}} and bind data in to html page{%if%}{%endif%} are easy to understand.

So choose one wisely and stick to it, after trying so many examples and tutorial finally I found some packages which are comfortable to work for me.

At starting I was too much confused between these packages and was unable to decide what to use.

For that I created separate projects for experimenting, which package is comfortable for me, and finally decided for my project, you might not agree with it you can use other packages.

This is where problem lies, which package to use and which to not, you will get some examples which use different packages to do some task, but may not work well together with alternate packages, this is where you will be wasting time, this is where you will get frustrated, that's why I have mentioned which packages iam using.

What are you going to do, about constant changes happening in nodejs and koa framework

This eBook may get absolute within 6 months, that's why I have created a website, which this book purchaser will get free access

After using this code you will be able to register on our website, where you will get free source code, and the questions about the code mentioned will be answered.

http://nodecode.info/

Some of the code may get absolute, but the latest code will be written on the website, we are developing many websites on nodejs technology, and it will be available free on the website.

In kindle edition, code written may not be seen clearly, so I will urge to visit my website and see the code.

What are you going to learn?

In this eBook, I will give you complete source code for website with basic features, login, registration, sessions, edit, and delete functions.

I will also explain how to do it and explain the code, if you get errors you can post it on our website and we will solve it.

These are the basic things for every web app, once you are able to code it, you can create any web app.

We have team of programmers who are developing next generation web apps, we will also be posting source code and discussing about it.

We are using

Node v4.4.7

Mongodb v3.2.3

Download and install node from https://nodejs.org/en/ according to your operating system

Download and install Mongodb from

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

Let's Start Coding

Iam doing a simple project with login, registration, edit profile, delete profile and view profile. With bootstrap 3 css using bootswatch theme. We are using simple html pages and for templating swig.

Note: If you are totally new then you have to look for , what packages and package versions you are using in project, don't get confuse in packages, there are so many packages to use,

Iam using Webstorm for developing web app

You can download it from here,

https://www.jetbrains.com/webstorm/download/

try trial version, and use its all features

Create new empty project in Webstorm with name loginRegitrationPract

Open terminal window at the bottom

Type following commands

npm init

This command is to create package.json file for project

Now this command will need some inserts and enters

Name: (loginRegitrationPract)loginregistrationpract // if you want to change then type the name and then press enter

Version:(1.0.0)0.0.1 // to change version type and enter

Description: koa v2 simple login registration with google project

Main:(index.js)app.js

Scripts: //note: just press enter if you don't want to fill any thing

And at last enter your package.json is ready

Now in project window you can see package.json file is created

```
{
  "name": " loginregistrationpract ",
  "version": "0.0.1",
  "description": " koa v2 simple login registration with google project ",
  "main": "app.js",
  "scripts": {
    "test": "echo ₩"Error: no test specified₩" && exit 1"
  },
  "keywords": [
    "koa",
    "node",
    "monk"
  ],
  "author": "Priya Patil",
  "license": "ISC"
}
```

This is how you file will look

Packages you need

- koa as a framework,
- monk and co-monk for Mongodb connection, co-monk for wrapper
- koa-route for routing
- koa-static to parse static page
- koa-views to render views
- swig for templating
- koa-convert, koa-generic-session for sessions
- koa-bodyparser to parse page data
- passport-local, koa-passport, passport-google-auth for login and registration and authentication

- co for wrapping generator function because koa v2 doesn't support for it

now in terminal window type command

```
>npm install koa@next --save
```

to install latest koa version 2, '--save ' will add koa version details to package.json file under dependencies.

Open package.json and you will see

```
"dependencies": {
  "koa": "^2.0.0-alpha.4",
}
```

//note: this the current koa version but not stable you can get all the latest packages using @next after packagename but are not stable versions

And in project solution window new folder is added 'node_modules' where all project dependencies are installed

project structure will be like this

- lib
  - db.js // to save database connection which we can access in other pages
- public
  - css,imges,etc.. // save all css files images logos here
- routes
  - homeRoutes.js // write down middleware module.export function for our routes
- views //all html pages here
  - home.html
  - login.html
  - layout.html
  - app.js //main application file₩
  - auth.js // authentication login registration code here
  - babel.app.js // to work async and await we have to use
  - package.json

Ok now what is this babel.app.js , for running async await functionality from koa v2 we have to run our project through babel

So we also need babel.js

Learn more about it from here

https://babeljs.io/

package.json

```
"dependencies": {
  "babel-core": "^6.13.2",
  "babel-polyfill": "^6.9.1",
  "babel-preset-es2015": "^6.13.2",
  "babel-preset-stage-0": "^6.5.0",
  "co-monk": "^1.0.0",
  "koa": "^2.0.0-alpha.4",
  "koa-bodyparser": "^3.2.0",
  "koa-convert": "^1.2.0",
  "koa-generic-session": "^1.11.3",
  "koa-passport": "^2.2.2",
  "koa-route": "^3.1.0",
  "koa-static": "^3.0.0",
  "koa-views": "^5.0.2",
  "monk": "^3.1.1",
  "passport-google-auth": "^1.0.1",
  "passport-local": "^1.0.0",
  "path": "^0.12.7",
  "swig": "^1.4.2"
}
```

You have to install all these dependencies with command

npm install <dependancyname> --save

As per our project structure create folders
lib,public,routes,views

And create app.js file under your main project directory.

Got the idea, now start your project ?

A basic Hello world program to make you happy, that your code actually works, Hurray

As per our project structure create folders lib, public, routes, views and create app.js file under your main project directory.

First we will create simple app.js to run without any view or template.

./app.js

//create application object

```
var koa = require('koa');
var app = new koa();
app.use(function (ctx){
      ctx.body = 'Hello World'
});

//this will listen your app on http://localhost:3000
app.listen(3000);
console.log('App listening on port 3000');
```

simple code to start your application
now in your terminal window type command

```
>node app
```

note: for koa v1 we type node --harmony app.js, but in v2 you don't have to use --harmony.

This will start your project and will show the message in terminal

```
App listening on port 3000
```

Now in a browser type localhost:3000 and you will see a hello word

Congrats, you have made your program work

Go Ahead add Some Real Code and Build your Application

Now you have been motivated, your hello world program worked, now we are going to add more functionality to our app.

Create db.js in your lib folder

./lib/db.js

```
var monk = require("monk"); //for mongodb connection
var wrap = require("co-monk"); // co-monk for wrapper
var db = monk("localhost/loginReg"); //connection to db loginReg

var users = wrap(db.get("users"));
// we are using users collection to save users details

//this will make users available in other pages
module.exports.users = users;
```

We are using monk and co-monk modules for database connections; we are using Mongodb in backend
We are exporting users to other modules so from any page it could be worked on.

./babel.app.js

//notes: it is compulsory to use babel or bluebird to run our project otherwise you will get error //running async await functionality

```
require("babel-core/register")({
    "presets": [
        "es2015",
        "stage-0"
    ]
});
require("babel-polyfill");
require('./app.js');
```

Now add the below code in App.js

./app.js

```
var koa = require("koa");
var app = new koa();
var route = require("koa-route");
```

```
var serve = require("koa-static");
var path = require("path");
var co = require("co");

var views= require("koa-views");// to render html swig template

app.use(views('views', {map:{html:'swig'}}));  // this will get our html swig
templates from views folder

// trust proxy
app.proxy = true;

// body parser
const bodyParser = require('koa-bodyparser');
app.use(bodyParser());

app.use(serve(__dirname + "/public"));

//routes*/
var homeRoutes = require("./routes/homeRoutes.js"); //getting js in variable

app.use(route.get("/", homeRoutes.showHome));//calling exported function

app.listen(3000);
console.log('Listening on http://localhost:3000');
```

Now add the code in homeRoutes.js

./routes/homeRoutes.js

//async function to render home.html page

```
module.exports.showHome = async (ctx) => {

    await ctx.render('home');
}
```

Now to run this app we have to use new command window because other is running by babel

```
>node --harmony babel.app.js
```

and run

this will run our project

copy this command and paste it in package.json under scripts

```
"scripts": {
  "start": "node --harmony babel.app.js",
  "test": "echo \"Error: no test specified\" && exit 1"
}
```

Now in terminal type command

```
>npm start
```

And your project is started. Before starting our project we have to create our view pages, I am using bootswatch for bootstrap css themes

https://bootswatch.com/

./views/layout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{%block title%}Title not set{%endblock%}</title>
    <link rel="stylesheet" href="/bootstrap.css">
    <link rel="stylesheet" href="/costum.css">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// --
>
    <!--[if lt IE 9]>
    <script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
    <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
</head>
<body>

<!—navbar for navigation we are already assigning links for login and logout
it might give error so comment the code if error -->


<nav class="navbar navbar-default">
    <div class="container-fluid">
```