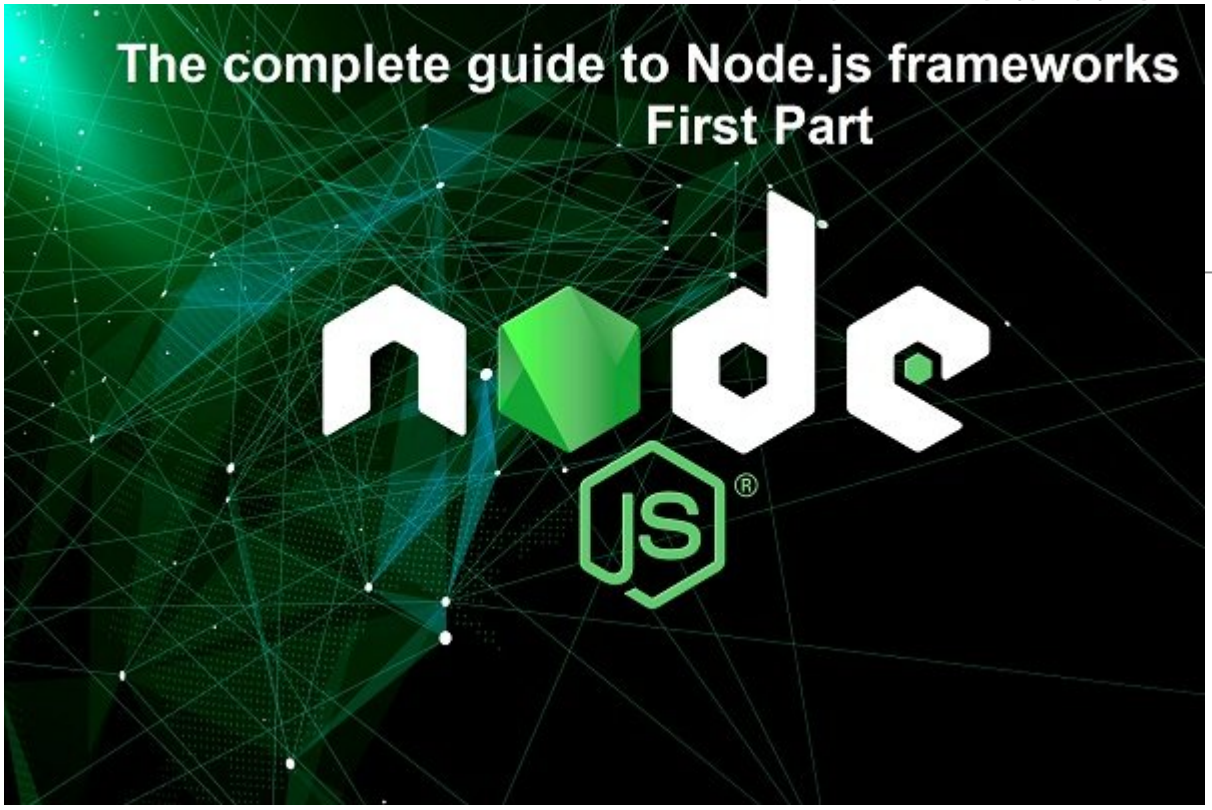
ویژه طرف‌داران جاوااسکریپت و طراحان وب

راهنمای جامع چهارچوب‌های برتر Node.js



Node.js یک محیط اجرایی بر پایه موتور جاوااسکریپت V8 کروم است که برای اجرای اپلیکیشن‌های محیط دسکتاپ و سرور بهینه‌سازی شده است. Node.js از یک مدل ورودی/ خروجی non-blocking رویدادمحور استفاده می‌کند که با کمترین تأخیر و بیشترین میزان بازدهی نسبت به سرورهای رشته‌ای مثل Apache, IIS و سرور جاوای متعارف به درخواست‌ها پاسخ می‌دهد. هرچند شما می‌توانید تنها از طریق کدنویسی Node.js یک وب سرور یا اپلیکیشن را پیاده‌سازی کنید، اما یک فریم‌ورک می‌تواند میزان کدی را که باید بنویسید تا حد قابل ملاحظه‌ای کاهش دهد. در این راهنما، قصد داریم فریم‌ورک‌های در دسترس برای توسعه‌دهندگان Node.js را در دو بخش مجزا به شما معرفی کنیم.

در ابتدا کار را با فریم‌ورک‌های مینیمالیستی شبیه Sinatra مثل Express آغاز می‌کنیم و به‌سمت فریم‌ورک‌های مستقل‌تر شبیه Rails مثل Sails.js پیش خواهیم رفت. در انتها به سراغ فریم‌ورک‌های کاملاً مستقل با کتابخانه‌های پایدار مثل Meteor خواهیم رفت. سرانجام، فریم‌ورک‌های REST API از قبیل LoopBack و چند کتابخانه دیگر که برای اهدافی خارج از دسته‌بندی اصلی ما قرار دارد (مثل IoT، ORM و تولید سایت‌های ایستا) را بررسی خواهیم کرد. توجه داشته باشید که این طبقه‌بندی‌ها به طور کامل از هم جداسازی نشده‌اند. چندین فریم‌ورک وجود دارد که می‌توان آن‌ها را متعلق به چند طبقه‌بندی دانست. پروژه‌های Node.js MVC بیشتری نیز وجود دارد که در فهرست ما گنجانده شده‌اند. هدف ما در معرفی این فریم‌ورک‌ها این است که بتوانید پروژه‌هایی را که ممکن است بتواند در زمان شما صرفه‌جویی کند را شناسایی کنید.

فریم‌ورک‌های MVC برای Node.js

MVC (سرنام Model-View-Controller) یک پارادایم در نظر گرفته شده برای پیکربندی شفاف عملکرد یک اپلیکیشن دسکتاپ یا وب است. مدل (Model) ساختار داده‌های پایه را مدیریت می‌کند.
نمایه (View) آن چیزی که به کاربر نشان داده می‌شود را مدیریت می‌کند و کنترل‌کننده (Controller) عکس‌العملی که در پاسخ به درخواست کاربر صادر می‌شود را مدیریت می‌کند. Rails یک فریم‌ورک وب MVC محور کامل است که در سال 2004 توسط دیوید هینمایر هنسون ساخته شد تا امکان ایجاد حضور وب در Ruby را فراهم کند. Rails فرض می‌کند که شما از یک مرکز داده استفاده می‌کنید و در پیکربندی مقادیر و مقیاس‌ها را به‌خوبی لحاظ کرده‌اید. فریم‌ورک‌های Node.js MVC شبیه Rails همان‌هایی هستند که از تمام قابلیت‌ها برخوردارند. Sinatra یک فریم‌ورک وب MVC محور با امکانات پایه و کم‌اهمیت‌تر است که در سال 2007 توسط بلیک میزرانی ساخته شد و در حال حاضر توسط کونستانتین هاسه به کار خود ادامه می‌دهد. Sinatra رویکردی متفاوت از Rails دارد و تنها شامل چیزهایی است که شما برای ساخت یک اپلیکیشن وب نیاز دارید و اساساً راه را برای قرار دادن اپلیکیشن شما در وب توسط یک DSL (سرنام Domain Specific Language) روی یک لایه Rack هموار می‌کند. Rack یک لایه

انتزاعی مبتنی بر Node.js EventEmitter است که راهی ساده را برای برخورد با پشتیبانی از کلاستر فراهم می‌کند. فریم‌ورک‌های Node.js MVC به شما اجازه می‌دهند در موارد لزوم اجزایی را به آن اضافه کنید. خیلی از فریم‌ورک‌های Node.js MVC شبیه Sinatra کار پیکربندی مقادیر را نیز انجام می‌دهند. به همین دلیل، متمایز کردن آن‌ها با فریم‌ورک‌های شبیه Rails همیشه هم به‌وضوح قابل انجام نیست.

مطلب پیشنهادی

یک محیط اجرایی چندسکویی
اکوسیستم Node.js چیست و چه قابلیت‌هایی دارد؟

فریم‌ورک‌های MVC شبیه Sinatra برای Node.js

Diet

Diet خود را یک فریم‌ورک وب Node.js کوچک و ماژولار می‌داند که برای ساخت سریع اپلیکیشن‌ها و API گسترش‌پذیر مناسب است. یک سرور پایه Diet بسیار شبیه به یک سرور پایه Express است.

```
// Create an app
  var server = require('diet')
  var app = server()
  app.listen('http://localhost:8000')

// When http://localhost:8000/ is requested, respond with "Hello World!"
  app.get('/', function($){
    $.end('Hello World!')
  })
```

فهرست 1

Diet از ساختاری برخوردار است که بدون نیاز به هرگونه ماژول یا پیکربندی اضافی از میزبانی مجازی پشتیبانی می‌کند. سرور Diet از توابع به‌عنوان سرورهای مجازی استفاده می‌کند و کافی است از طریق پورت‌های مختلف به آن‌ها دسترسی پیدا کنیم. مسیریابی در Diet نه‌تنها با توابع ناشناس مثل app.get() در فهرست 1 اداره می‌شود، بلکه می‌توان این کار را از طریق یک مسیر middleware همانند فهرست 2 نیز انجام داد.

```
// Register middleware functions for the upload path
  app.post('/upload/picture', upload, crop, save, finish)
```

فهرست 2

Node.js در زمان ساخت یک سرور (HTTP(s با استفاده از http.createServer() به طور پیش‌فرض از دو نشان‌وند request و response برخوردار است. Diet این دو مقدار را به یک مقدار تبدیل می‌کند که توسط علامت $ ارائه می‌شوند. در فهرست 1 در app.get() شما مشاهده می‌کنید که تنها یک مقدار در نشان‌وند این تابع که درخواست‌های get را در مسیر ریشه اداره می‌کند وجود دارد. Diet از ماژول‌های Node.js نیز پشتیبانی می‌کند و می‌تواند از آن‌ها به‌عنوان middleware استفاده کند.

Express

Express یک فریم‌ورک اپلیکیشن وب Node.js مختصر و انعطاف‌پذیر است که برای ساخت صفحات اپلیکیشن‌های وب مجموعه‌ای از قابلیت‌های قدرتمند را ارائه می‌کند. Express API اپلیکیشن‌های وب، درخواست‌ها و پاسخ‌های

HTTP، مسیریابی و middleware را مدیریت می‌کند. از نسخه Express 4.x میان‌افزارهای پشتیبانی شده برای Express در تعدادی مخزن جداگانه قرار گرفته است که فهرست آن در مخزن Connect قرار دارد. چندین فورک و افزونه برای Express در نظر گرفته شده است که شامل Locomotive, Hapi و Koa است. Koa توسط یکی از مشارکت‌کنندگان اصلی Express ساخته شده است. Express نسبت به سایر فریم‌ورک‌ها از اجتماع و گروه‌های پشتیبانی بزرگ‌تری برخوردار است و با خیلی از ابزارها و فریم‌ورک‌های دیگر برای ساخت وب سرورها در Node.js ترکیب می‌شود.

```
// create an express application

const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello world!')
})

app.listen(3000, function () {
  console.log('Example app listening on port 3000!)
})
```

فهرست 3

مطلب پیشنهادی

تکنیک‌های کاربردی ES6
10 ویژگی جدید و جذاب جاوااسکریپت برای عاشقان وب

Flatiron

Flatiron بخشی از مجموعه ابزار Nodejitsu Node است. نویسندگان این فریم‌ورک آن را برای دو منظور در نظر گرفته‌اند. ابتدا، یک ابتکار عمل برای ساخت مجموعه‌ای از ابزارهای جدا شده با کیفیت و عملکرد بالا. دوم، یک فریم‌ورک توسعه اپلیکیشن وب کامل که با گردآوری این ابزار در کنار یکدیگر توسعه اپلیکیشن‌های وب را ساده‌تر کند. Flatiron به این دلیل در دسته فریم‌ورک‌های شبیه Sinatra قرار گرفته که تمام کاری که شما برای استفاده از آن در یک وب سرور باید انجام دهید این است که درخواست (require) را مشخص کنید، یک اپلیکیشن تعریف کنید، از پلاگین‌های http استفاده کنید، چند مسیر تنظیم و اپلیکیشن را آغاز کنید.
سایر اجزای این مجموعه نیز کارایی Flatiron را افزایش می‌دهند. برای مثال Broadway یک پلاگین ساده API را ارائه می‌کند که می‌تواند جایگزینی برای معکوس کردن کنترل ثبت نام استفاده شده توسط سایر فریم‌ورک‌های Node MVC باشد. Union یک کرنل میانی برای بافر و استریم است که با Connect سازگار است. Union بخشی است که پلاگین http را ارائه می‌کند.

```
// create a flatiron application

var flatiron = require('flatiron'),
  app = flatiron.app;

app.use(flatiron.plugins.http);

app.router.get('/', function () {
  this.res.writeHead(200, { 'Content-Type': 'text/plain' });
```

```
  this.res.end('Hello world!\n');
});

app.start(8080);
```

فهرست 4

Hapi

Hapi یک فریم‌ورک با امکانات استفاده و پیکربندی ساده است که از اعتبارسنجی ورودی‌ها، کش، احراز هویت و سایر امکانات ضروری برای ساخت یک وب‌سایت و خدمات تحت وب پشتیبانی می‌کند. Hapi توسعه‌دهندگان را قادر می‌سازد تا از طریق رویکردهای تمام ماژولار و سفارشی روی نوشتن اپلیکیشن‌های چند بار مصرف تمرکز کنند. Hapi توسط آزمایشگاه‌های والمارت توسعه پیدا کرد و یک انتخاب ایده‌آل برای گروه‌ها و پروژه‌های بزرگ است. Hapi در اصل به‌عنوان بخشی از Express ساخته شد، اما بعد کار خود را به‌صورت مستقل ادامه داد. همان گونه که سازندگان آن اعلام می‌کنند، Hapi با این شعار ساخته شد که «پیکربندی بهتر از کدنویسی است و این منطق تجاری باید از لایه انتقال جدا باشد». در مثال زیر، مشاهده می‌کنید که پیکربندی مسیرهای سرور چقدر واضح و مشخص در کد قرار گرفته است.

```
// create a hapi server

var Hapi = require('hapi');
var server = new Hapi.Server(3000);

server.route([
  {
    method: 'GET',
    path: '/api/items',
    handler: function(request, reply) {
      reply('Get item id');
    }
  },
  {
    method: 'GET',
    path: '/api/items/{id}',
    handler: function(request, reply) {
      reply('Get item id: ' + request.params.id);
    }
  },
```

فهرست 5

Koa

Koa یکی از جدیدترین فریم‌ورک‌های وب است که توسط گروه سازنده Express طراحی شده است، اما مستقل از کدهای Express کار می‌کند. هدف از ساخت Koa امکان ایجاد اپلیکیشن‌های وب و API کوچک‌تر، مفهوم‌تر و با پایه‌ریزی قوی‌تر بوده است. Koa به‌جای فراخوانی‌های Node.js از مولدهای ES6 برای میان‌افزار استفاده می‌کند. کد زیر یک اپلیکیشن Hello, World نوشته شده با Koa آورده شده است که از یک مولد که با yield next کنترل را به مولد بعدی منتقل می‌کند، استفاده کرده است.

```
var koa = require('koa');
var app = koa();
```

```
// x-response-time
app.use(function *(next){
  var start = new Date;
  yield next;
  var ms = new Date - start;
  this.set('X-Response-Time', ms + 'ms');
});

// response
app.use(function *(){
  this.body = 'Hello World';
});

app.listen(3000);
```

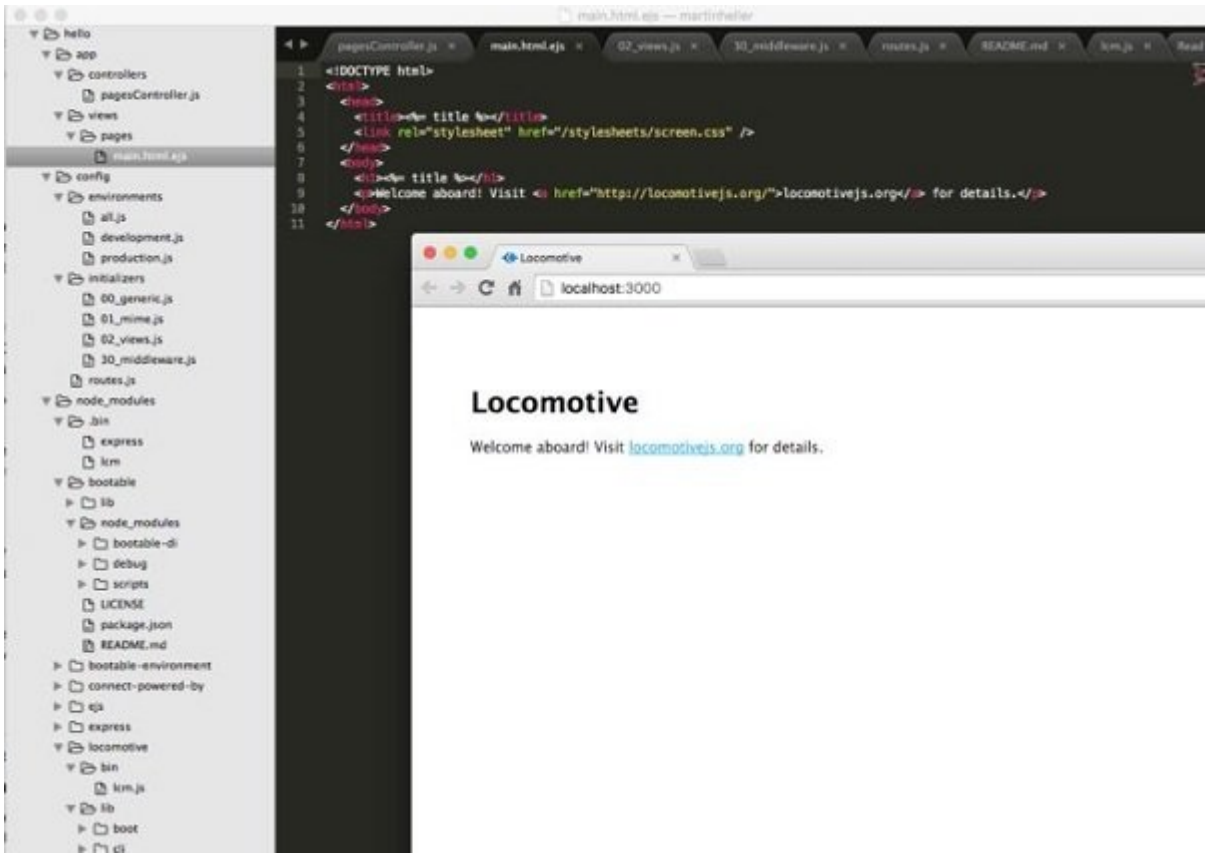
فهرست 6

بین مولدهای middleware استفاده شده توسط Koa و فراخوانی‌هایی که توسط Express و Connect استفاده می‌شود تفاوت وجود دارد. پیاده‌سازی Connect به‌سادگی کنترل را از طریق مجموعه‌ای از توابع منتقل می‌کند، در حالی که yield در Koa فروکش می‌کند و سپس چرخه کنترل دوباره اوج می‌گیرد. در فهرست 6، x-response-time مولد پاسخ را با عبارت yield next پوشش داده است.

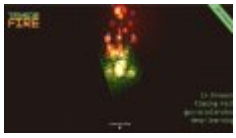
Locomotive

Locomotive یک فریم‌ورک وب مخصوص Node.js است که از الگوهای MVC، مسیرهای RESTful و convention over configuration پشتیبانی می‌کند و می‌تواند با هر نوع مرکز داده و موتور الگو یکپارچه شود. Locomotive بر پایه Express و Connect ساخته شده است.
همان گونه که در شکل 1 مشاهده می‌کنید، Locomotive بعضی از ساختارهای شبیه Ruby-on-Rails را به Express اضافه می‌کند. نمایه‌های Locomotive اغلب در فایل‌های جاوااسکریپت (html.ejs) جاسازی می‌شوند، اما Locomotive از Jade و سایر موتورهای الگوی سازگار با Express نیز پشتیبانی می‌کند. عملکرد REST توسط مسیرها کنترل می‌شود. شما می‌توانید با Locomotive از هر نوع مرکز داده و لایه ORM که مایل باشید استفاده کنید.

شکل1



مطلب پیشنهادی



کاربردی‌ترین کتابخانه یادگیری ماشینی
یادگیری ماشینی از طریق جاوااسکریپت به درون مرورگرهای کاربران آمد

Total.js

Total.js یک فریم‌ورک سمت سرور کامل برای Node.js است که مثل Laravel پی‌اچ‌پی یا Django پایتون تماماً با جاوااسکریپت نوشته شده است. سکوی Total.js متشکل از مجموعه‌ای از کتابخانه‌ها، بسته‌ها و محصولات تکمیلی است که توسط خود Total.js ساخته شده‌اند. با استفاده از قطعه کد فهرست 7 می‌توان یک وب‌سرور ابتدایی Total.js را پیاده‌سازی کرد.

```
require('total.js');

F.route('/', function() {
    this.plain('total.js is really good!');
});

F.http('debug');
```

فهرست 7

فریم‌ورک‌های MVC شبیه Rails برای Node.js
Adonis

Adonis یک فریم‌ورک MVC مخصوص Node.js است که بر اساس موارد کاربرد عملی ساخته شده است. این

فریم‌ورک یک ابزار CLI را برای چهارچوب‌بندی و تولید یک پروژه فراهم می‌کند. از جمله ویژگی‌های Adonis می‌توان به اجرای الگوی طراحی رکورد فعال، لایه احراز هویت به همراه سشن‌ها، JWT، اعتبارسنجی ساده، نشانه‌های API شخصی و پیاده‌سازی کنترل‌کننده‌ها به‌عنوان کلاس‌های ES2015 اشاره کرد. مولدهای ES2015 فراخوانی‌های اضافی مرسوم در جاوااسکریپت را حذف می‌کند. فهرست 8 تمام کاربران را از مرکز داده استخراج و در قالب JSON ارائه می‌کند.

```
const Route = use('Route')
const User = use('App/Model/User')

Route.get('/', function * (request, response) {
  const users = yield User.all()
  response.json(users)
})
```

فهرست 8

CompoundJS

فرمول پشت CompoundJS شامل Express + ساختار + افزونه‌ها است. ساختار همان استاندارد صفحه‌بندی و دایرکتوری‌ها است و افزونه‌ها ماژول‌های Node.js هستند که کارایی این فریم‌ورک را افزایش می‌دهند. هدف CompoundJS تأمین یک رابط واضح و سازمان‌یافته برای توسعه اپلیکیشن‌های سازگار با Express است. این به این معنا است که هرچه با Express کار می‌کند با CompoundJS هم کار خواهد کرد. شما می‌توانید چهارچوب اپلیکیشن‌های CompoundJS را با CLI تولید کنید.

```
npm install compound -g
compound init todo-list-app
cd todo-list-app && npm install
node .
```

فهرست 9

سایت به طور پیش‌فرض به http://localhost:3000/ گوش می‌کند. شما می‌توانید با فرمان compound generate scaffold بر مبنای مدل‌ها چهارچوب درست کنید.

Geddy

فریم‌ورک Geddy به همان شیوه Rails یکMVC را برای Node.js پیاده‌سازی می‌کند. با همان ساختار دایرکتوری، قابلیت باز کردن کنسول REPL در اپلیکیشن و تولیدکننده اسکریپت که شما می‌توانید برای ساخت اپلیکیشن‌ها از آن استفاده کنید. چهارچوب‌بندی را می‌توان با استفاده از قالب‌های Mustache، Handlebars، Jade، EJS و Swig انجام داد. فرمان geddy jake می‌تواند وظایف Jake (JavaScript make) گوناگونی را در اپلیکیشن انجام دهد. این امر برای وظایف جانبی مانند تست، پیاده‌سازی توسعه مرکز داده و فهرست‌بندی مسیرها مفید است.

Kraken

یک پروژه منبع باز PayPal امن با لایه گسترش‌پذیر که امکانات Express را با فراهم کردن ساختار و ضوابط مثل Locomotive گسترش می‌دهد. هرچند Kraken ستون اصلی فریم‌ورک محسوب می‌شود، از این ماژول‌ها نیز می‌توان به‌صورت جداگانه استفاده کرد:

Lusca (امنیت)، Kappa (NPM Proxy)،
Makara (LinkedIn Dust.js i18N) و
Adaro (LinkedIn Dust.js Templating).

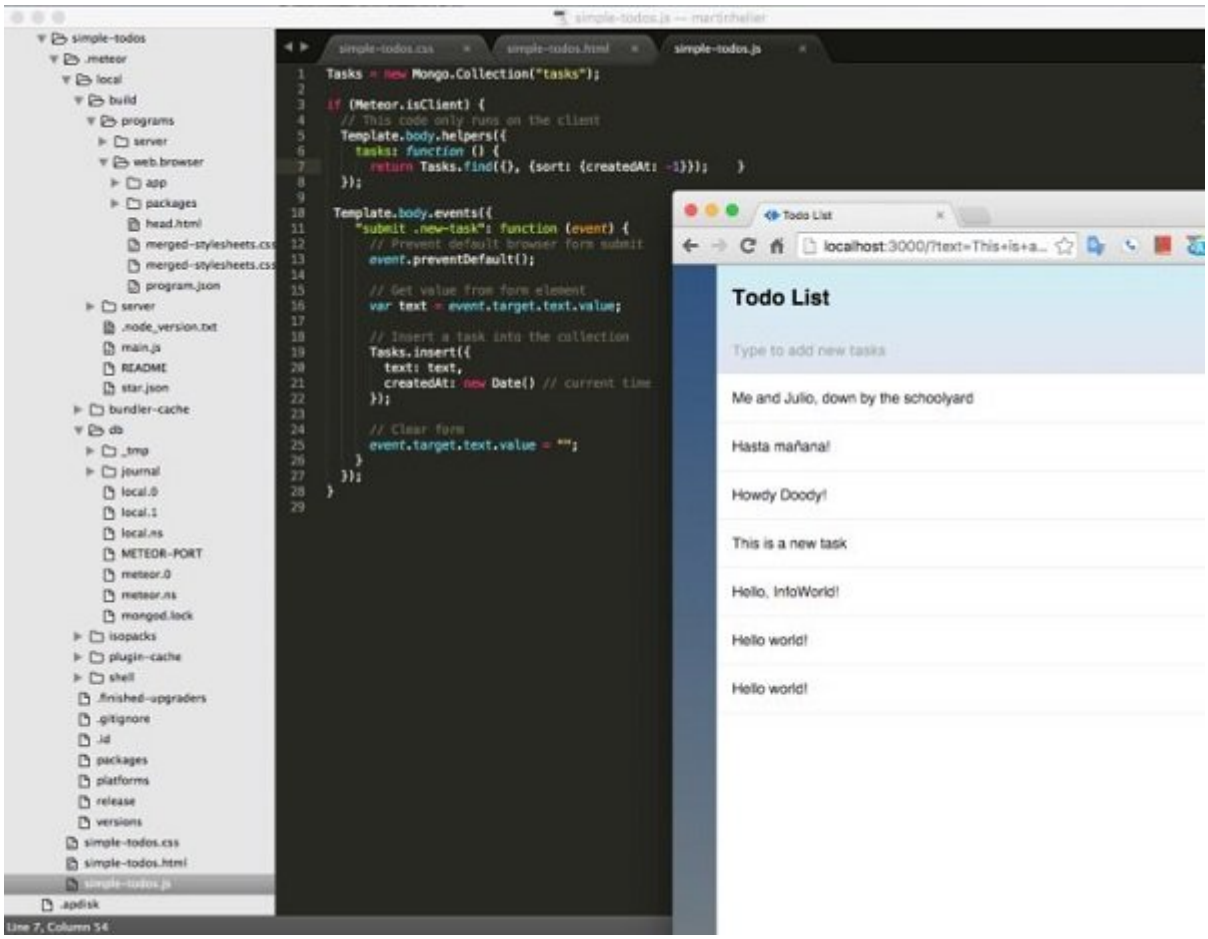
همان گونه که شکل 2 مشاهده می‌کنید، Kraken برای تولید پروژه‌ها به yo متکی است. مثل Locomotive این فریم‌ورک نیز پروژه‌های خود را به‌شیوه مرسوم Rails سازماندهی می‌کند.

شکل 2

Meteor

Meteor یک سکوی اپلیکیشن کامل است که به شما امکان می‌دهد تا با یک روش ساده تماماً با جاوااسکریپت و از طریق یک کد پایه، اپلیکیشن وب یا موبایل بسازید. علاوه بر تبادلات از طریق HTML، این فریم‌ورک داده را برای پردازش از سرور به کلاینت ارسال می‌کند. علاوه بر اجرای مستقل، Meteor می‌تواند با Angular و React نیز برای پشتیبانی از الگوی MVC یکپارچه شود. با وجود اینکه این فریم‌ورک روی Node.js ساخته شده است و از Handlebars, Blaze و قالب‌های Jade پشتیبانی می‌کند، اما نمی‌توان Meteor را شبیه Express دانست. Meteor این امکان را برای شما فراهم می‌کند تا برای پروژه خود نمونه‌سازی و کدهایی با قابلیت پشتیبانی از تمام سکوها (وب، اندروید و iOS) ایجاد کنید. قابلیت یکپارچه شدن با MongoDB را دارد و از پروتکل داده توزیع شده (DDP) استفاده می‌کند. در سمت کلاینت، Meteor به jQuery وابسته است و می‌توان آن را با کتابخانه JavaScript UI استفاده کرد.
Meteor توسط گروه توسعه Meteor که یک استارت‌آپ سرچشمه گرفته از Y Combinator است تولید شده است. این فریم‌ورک به اندازه‌ای جا افتاده است که چندین کتاب آموزشی درباره آن نوشته شده است.
Meteor به‌تنهایی یک نرم‌افزار منبع باز رایگان است، اما گروه Meteor آن را با دریافت حق عضویت Galaxy DevOps پولی کرده‌اند که شامل فضای سرور AWS و پشتیبانی از Meteor می‌شود. (شکل 3)

شکل 3



Nodal

Nodal خود را یک سرور وب Node.js معرفی می‌کند که برای ساخت سرویس‌های API بهینه‌سازی شده است. این سرور وب فریم‌ورک مستقل، واضح و توسعه‌پذیر مربوط به خود را دارد که فلسفه آن بیشتر شبیه به Rails و Django است. می‌توان PostgreSQL را به طور مستقل یا ماژول مرکز داده با Nodal یکپارچه کرد، از آن به‌شکل ORM کوئری گرفت، در مسیریابی آن از Regex استفاده کرد و ابزار CLI را برای ساخت مدل‌ها و کنترل‌کننده‌ها به کار گرفت. برای راه‌اندازی یک سرور Nodal ابتدا باید Nodal را نصب و سپس یک سرور جدید را آماده و راه‌اندازی کنید.

مطلب پیشنهادی

Napa.js قرار است عملکردی شبیه به C++ داشته باشد
مایکروسافت امکان محاسبات چندرشته‌ای را برای انجام وظایف سنگین محاسباتی به Node.js اضافه کرد

Sails

Sails کار ساخت اپلیکیشن‌های Node.js سفارشی و حرفه‌ای را ساده می‌کند. این فریم‌ورک طراحی شده است تا الگوی آشنای MVC را شبیه‌سازی کند که می‌تواند نیازهای اپلیکیشن‌های مدرن مثل API داده‌محور با امکان بسط‌پذیری و معماری مبتنی بر خدمات را نیز برآورده کند. استفاده از آن به‌ویژه برای ساخت اپلیکیشن‌های چت، داشبوردهای پویا و بازی‌های چندنفره ایده‌آل است، اما می‌توانید از آن برای ساخت هرگونه اپلیکیشن تحت وب دیگر نیز استفاده کنید. Sails از WebSocket پشتیبانی می‌کند و به طور خودکار پیام‌های سوکت را به مسیرهای

اپلیکیشن شما ارسال می‌کند. Sails روی Express و Socket.io ساخته شده است و برای ORM خود از Waterline استفاده می‌کند. Waterline انواع مختلفی از شیوه‌های ذخیره‌سازی داده و مراکز داده مختلف (SQL و NoSQL) را پشتیبانی می‌کند. Sails به‌گونه‌ای طراحی شده تا با تمام فریم‌ورک‌های وب کاربردی مثل Angular و React یا دستگاه‌های موبایل مثل iOS و اندروید سازگار باشد. تاکنون سه کتاب درباره Sails به چاپ رسیده است.

ThinkJS

ThinkJS یک فریم‌ورک MVC است که از قابلیت‌های ES6 (تابع generator) و ES7 (async و await)، Babel، TypeScript، کتابخانه‌های WebSocket، انواع مختلفی از سشن و کش و مراکز داده MySQL, MongoDB, PostgreSQL و SQLite پشتیبانی می‌کند. EJS, Jade, Swig و موتورهای الگوی Nunjucks نیز توسط این فریم‌ورک پشتیبانی می‌شوند. سازنده مدعی است که ThinkJS قابلیت و پیچیدگی Sails و عملکرد قدرتمند Express یا Koa را یک جا جمع کرده است.

برای مطالعه قسمت دوم این مقاله روی لینک زیر کلیک کنید

مطلب پیشنهادی

مقاله‌ای ویژه طرف‌داران جاوااسکریپت و طراحان وب
راهنمای جامع چهارچوب‌های برتر Node.js (بخش دوم)

منبع:
Info World
تاریخ انتشار:
07 آذر 1396

نشانی منبع:
https://www.shabakeh-mag.com/workshop/10740/%D8%B1%D8%A7%D9%87%D9%86%D9%85%D8%A7%DB%8C-%D8%AC%D8%A7%D9%85%D8%B9-%DA%86%D9%87%D8%A7%D8%B1%DA%86%D9%88%D8%A8%E2%80%8C%D9%87%D8%A7%DB%8C-%D8%A8%D8%B1%D8%AA%D8%B1-nodejs