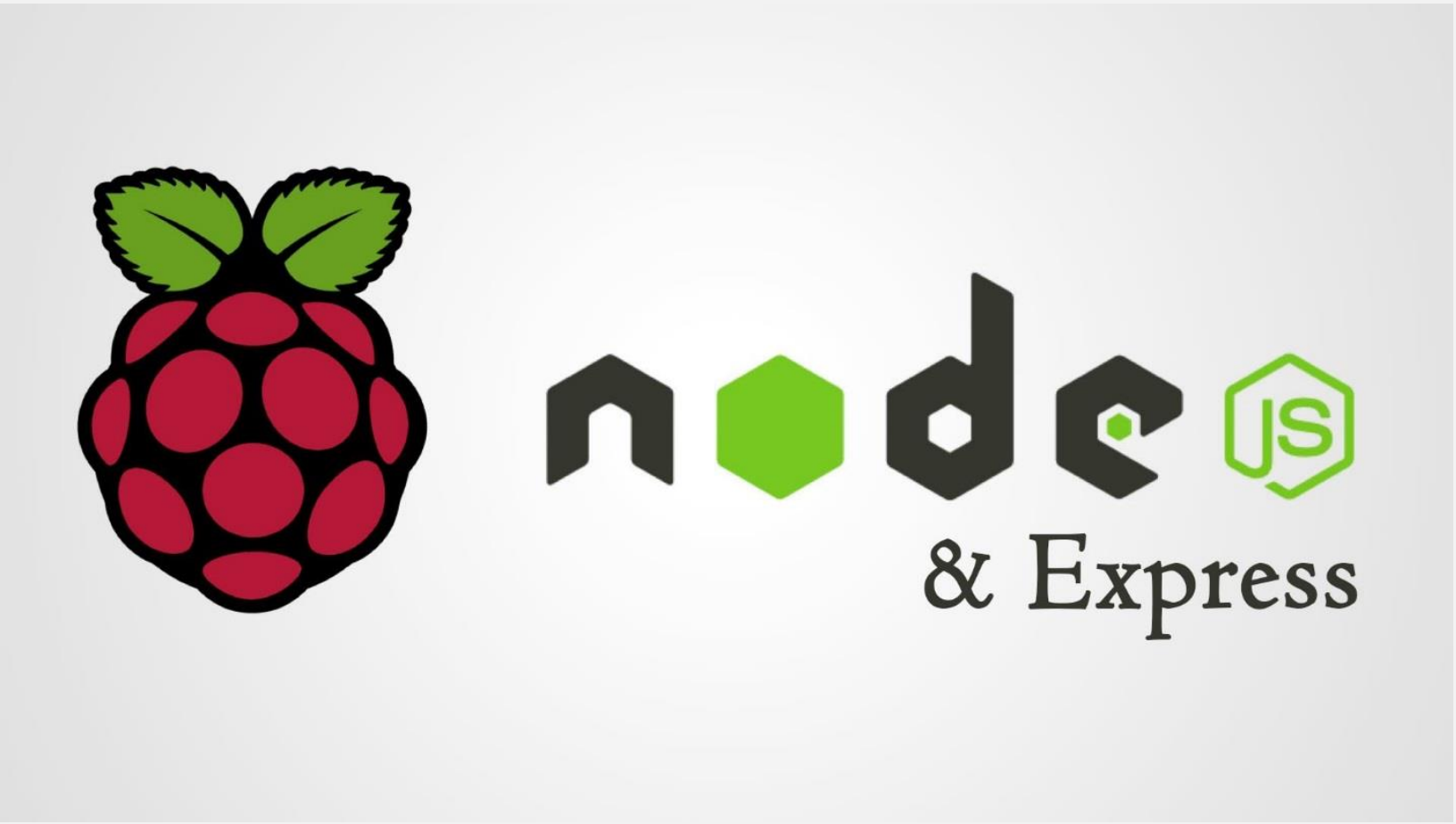


تالیف: حسن تفرشی

فهرست

فهرست	صفحه
درباره کتاب	3
ملاحضاتی جهت آگاهی خوانندگان	3
ساختار	5
جاوا اسکریپت وNode.js	6
جاوا اسکریپت و شما	6
جاوا اسکریپت سمت سرور	7
ساختن یک اپلیکشین کامل بوسیلهNode.js	8
پشته اپلیکشین	8
ساخت یک پشته اپلیکشین	9
آنالیز کد HTTP سرور	10
چگونه تابع سرویس https را پاس می دهد	12
Node.jsوexpress	14
اجرا کردن پروژه باexpress	14
یک اپ ساده باexpress	15
expressو نمایش	18
JADEچیست	19
HTMLوJADE	19
NODE.JSوJADE	20
ساخت یک اپ ساده با express	20
یک دایرکتوری عمومی برای اپ	23
مولد خودکار app در express	27
قدرت دادن express با خاصیت میان افزاری	29
Expressقدرت گرفته از ماژولNode	35
درخواست log به اپ	38
استفاده از یک فایل پیکربندی شده	40
Setو Get اپلکیشن	41
Environmentمتفاوت اکسپرس	42
منابع	43

درباره کتاب :

این کتاب با هدف آشنایی شما با node.js و درباره مواردی که احتیاج دارید درباره توسعه اپلیکشن Node.js نیاز دارد تالیف شده است و این متن مشابه آموزش های "Hello World" می باشد .

نکته :کد های در این کتاب در نسخه های ۰,۸,۸ به بالا Node.js به درستی کار می کند

ملاحضاتی جهت آگاهی خوانندگان :

این کتاب مناسب افرادی است که پیشتر از خواندن این کتاب آشنای کافی با زبانی object-oriented نظیر PHP , Python , Ruby یا Java و همچنین حداقل تجربه ی در زمینه کار با JavaScript را داشته باشند و اگر پیشتر از این هیچ آشنائی با Node.js ندارید این کتاب مناسب شما می باشد .

لازم است خواننده پیشتر از این در جهت توسعه با زبان های یاد شده بالا تجربه ای داشته باشد و همچنین با ساختار های Object-oriented آشنای داشته باشد زیر این متن به توضیح جنس متغییر ها یا کنترل ساختمان و توابع مسائل نظیر به این نمی پردازد . جهت درک متن این کتاب نیاز است شما با ساختارهای یاد شده پیشتر از این آشنائی داشته باشید .

به هر حال همانطور که مستحضر هستید ساختار توابع (functions) و شئی (object) در جاوا اکسریپت با اکثر زبان ها متفاوت است و در متن پیش رو در این خصوص جزئیات بیشتری ارائه خواهد شد.

ساختار

هدف این است که خواننده عزیز بعد از اتمام این کتاب توانی خلق یا به عبارت دیگری تولید یا به عبارت عامیانه توانی کدنویسی کامل یک اپلیکشین تحت که به کاربرها این اپلیکشین این اجاره را می دهد که فایل ها خود را در این اپلیکشین تحت وب بارگزاری کنند و در آموزش قدم به قدم تا رسیدن به خلق کامل این اپلیکشن به توضیح قسمت به قسمت کدها می شود .

حقیقتا همانطور که شما فکر می کنید قرار نیست این اپلیکیشن ساده دنیا را تغییر دهد اما ما همانطور که در پارگراف بالا یاد شده است این فقط کد نوشتن نیست ولی برای شروع کافی است . شما در طول این کتاب خواهید دید این اپلیکشین ساده تحت وب چه چیزهای در خصوص ساختار framework برای شما در Node.js را روشن خواهد کرد.

پیشتر از شروع می ببینیم که تفاوت توسعه دادن جاوا اسکریپت در Node.js با توسعه دادن جاوا اسکریپت در یک مرورگر (Browser) .

در ادامه ادامه به آموزش روش قدیمی و قدم به قدم به نام "Hello World " که بسیار پرثمره است خواهیم پرداخت و همچنین به معرفی Express می پردازم .

جاوا اسکریپت و Node.js

جاوا اسکریپت و شما

اگر شما هم مثل من توسعه را از طریق HTML را سال های پیش شروع کردید شما در سالهای قبل با زبانی آشنا شدید که به آن جاوا اسکریپت می گفتند آشنا شدید که کارهای بسیار باحال و دوست داشتنی انجام می داد و همچنان به این وظیفه خود ادامه می دهد اما شما فقط برای مسائل خیلی مقدماتی و تعاملی در توسعه وب ها استفاده می کردید .

حتمی شما نیز می خواستید یک حرکت واقعی یا بصورت دقیق یک توسعه واقعی با جاوا اسکریپت انجام دهید و احتمالا می خواستید بدانید چگونه می توان یک وب سایت های پیچیده را از این طریق ساخت و همانطور که شما کد نویسی و توسعه با زبان های نظیر جاوا و پی اچ پی و ... را شروع و پایان رساندید . گوشه چشمی همچنان در جهت توسعه به JavaScript داشته اید و شما دید به وسیله جی کوئری [1]بصورت پیشرفته تر از جاوا اسکریپت می توان عمل کرد اما این ساختار در حقیقت بیشتر درباره window.open() می باشد.

به هرحال در زمان استفاده جی کوئری در پروژه ها همه چیز خوب است اما در پایان پروژه شما همچنان یک کاربر هستید که از جاوااسکریپت استفاده می کنید **نه یک توسعه دهند جاوا اسکریپت** [2]

الان که Node.Js در سمت سرور در اختیار می باشد . به نظر شما باحال نیست ؟ جاوا اسکریپت سمت سرور!

همه چیز کامپیوتر و آی تی به زمان بر می گردد و بروز شدن ، همانطور که شما دنبال بروز کردن اطلاعات هستید باید بدانید ، نوشتن اپلیکیشن Node.js یک داستان است و متوجه بودن این امر که چرا شما باید با جاوااسکریپت اپلیکیشن بنویسید داستان متفاوتی است .

طریقه نوشتن اپلیکشین های Node.js به شکلی می باشد که شما فقط احساس نمی کنید که در حال استفاده جاوااسکریپت هستید در حقیقت شما در حال توسعه دادن Node.js می باشید و به عبارت صحیح تر **شما توسعه دهنده هستید نه یک کاربر.**

[1] JQuery
[2] JavaScript Developer

در اینترنت منابعی مختلفی در خصوص Node.js و جاوا اسکریپت وجود دارد اما در بسیار موارد داکیومنت و متون به تنهایی کافی نیستند و در حقیقت شما در یادگیری مسئله ای نیاز به راهنما دارید .

جاوا اسکریپت سمت سرور

از زمان پیدایش جاوا اسکریپت تا به امروز در سمت مرورگر قابلیت اجرا داشته است اما این فقط محتوا است , جاوا اسکریپت یک زبان کامل است. این متن به شرح این موضوع می پردازد که چگونه شما می توانید به تعریف جاوا اسکریپت سمت سرور بپردازید. Node.js به شما اجازه می دهد جاوا اسکریپت را در خارج یک مرورگر اجرا کنید.

Node.js دو موضوع است اول : یک محیط زمان اجرا و یک کتابخانه [۳] .

شما نیاز دارید که Node.js را نصب کنید در خصوص نصب به اینجا ر کلیک کنید

"Hello World"

خوب حالا بریم سراغ اولین آپ که با Node.js خواهیم نوشت این اپ سلام دنیا یا عبارت معروف " Hello world " نام دارد .

ایدتور مورد علاقه خود را باز کنید و فایل با نام helloworld.js بسازید .

قصد داریم از طریق STDOUT عبارت "Hello world" را برگردانیم یا به عبارت بهتر آن را چاپ کنیم . کد زیر را در فایل ذخیره کنید

```
console.log("Hello World");
```

حالا کد بالا را از طریق Node.js با فرمان زیر اجرا کنید :

```
node helloworld.js
```

شما در خروجی حتمی باید Hello world را مشاهده کنید. من هم با شما موافقم این اپ بسیار ساده و خسته کننده است حالا بیاد یک چیز یا به عبارت تخصص یک اپ واقعی تری بنوسیم .

[3] library

ساختن یک اپلیکشین کامل بوسیله Node.js

به سراغ اپلیکشین آپلود برویم

شما با جستجو در گوگل و دیدن سورس های دیگر نیز می توانید به این هدف که ساختن یک اپلیکیشن آپلود هست برسید اما این آن چیزی نیست که ما در اینجا قصد داریم به آن برسیم . هدف ما فهمیدن ساختار و روش کار کردن با node.js می باشد.

پشته اپلیکشین

- برای اجرا وب پیج ها به ساختار HTTP سرور نیاز داریم
- سرور باید توانای پاسخ گوئی در خواست های متفاوت از آدرس های متفاوت را داشت باشد .
- ما در حقیقت به request handler در سرور نیاز داریم در جهت اجرا درخواست های وارده به سرور .
- سرور باید توانای پشتیبانی از متد دیتا پست (Post)جهت هندل کردن درخواست ها داشته باشد در حقیقت نیاز به یک request data handling داریم

پیشتر یک لحظه فکر کنیم چگون می توان یک پشته با php ساخت . این موضوع یک راز نیست و با نصب آپاچی سرور این امر با کمک mod_php5 قابل انجام است.

در حقیقت ما نیاز داریم که سرور توانای دریافت درخواست های http از صفحات وب را داشته باشد . این امر ممکن نمی شود بوسیله خود php .

بوسیله node.js مسئله کمی متفاوت است . ما فقط به اجرا اپلیکشین نمی پردازیم بلکه به اجرا تمامی HTTP سرور نیز می پردازیم . در این خصوص اپلیکشن وب ما و وب سرور در بصورت پایه ای یکی می شوند .

ممکن است به نظر بیاد برای عملی شدن امر بالا کارهای زیاد باید انجام داد اما ما خواهیم دید بوسیله node.js یک لحظه بیشتر طول نمی کشد .

خوب بریم سراغ اجرا قسمت اول پشته در HTTP سرور.

ساخت یک پشته اپلیکشین

در سمت سرور

اول فایل اصلی که می خواهیم اپلیکشین ما را شروع کنید و یک ماژول فایل که قرار است کد HTTP سرور در آن قرار گیر را می سازیم.

پیشنهاد می شود بصورت استاندارد اسم فایل اصلی را index.js بگذارید و اسم فایل ماژول سرور را server.js انتخاب کنید.

فایل server.js ایجاد شده را در مسیر اصلی دایرکتوری پروژه خود قرار دهید و کد زیر را در درون آن قرار دهید :

```
var http = require("http");

http.createServer(function(request, response) {
response.writeHead(200, {"Content-Type":
"text/plain"});
response.write("Hello World");
response.end();
}).listen(8888);
```

به همین سادگی ! شما یک HTTP سرور ساختید . حالا برای اثبات صحت کار آن را اجرا و تست کنید . اول , اسکریپت اجرای node.js را اجرا کنید :

```
node server.js
```

حالا مرورگر خود را باز کنید و آدرس http://localhost:8888/ را وارد کنید شما خواهید دید که نوشته Hello World“ ” به نماش در آمده است

جالب بود نه ؟ حالا صحبتی که مطرح می شود این است که چگونه پروژه های آینده خود را سازماندهی کنیم .حتمی به این موضوع خواهیم پرداخت در فوصول آینده .

آنالیز کد HTTP سرور

حالا وقت اینه که ببینم دقیقا چه اتفاقی در کد قسمت قبل افتاده است . خط اول درخواست یک ماژول از جنس http شده بوسیله node.js و نام این متغییر را http گذشته ایم .
در خط بعد یکی از توابع ماژول http را فراخوانی کردیم که اسم آن تابع createServer می باشد این تابع یک شئی (object) را برای ما در دسترس قرار می دهد این شئی شامل یک متد به نام listen می باشد . متد listen یک مقدار عددی را بعنوان وردی می گیرد که همان شماره پورت برای http سرور ما جهت منتظر بودن فرمان می باشد.

برای لحظه ای توضیحات قسمت را در پس زمینه ذهنتان حفظ کنید و قسمت http.createSever توجه داشته باشید .
ما می توانسیتم کد شروع گرفتن سرویس را بنویسم و منتظر پاسخ از پرت ۸۸۸ باشیم مانند کد زیر

```
var http = require("http");
var server = http.createServer();
server.listen(8888);
```

این کد می تواند یک سرویس http را شروع و منتظر پاسخ از پورت 8888 باشد و هیچ کار دیگری نیز انجام ندهد (نه زمانی که پاسخی یا درخواستی دریافت می کند)
کد اول خیلی بهتر و جالب تر است (و اگر شما در پس زمینه از بیشتر یک زبان مانند php نیز استفاده کرده باشید) . قسمتی از تابع را تعریف کرده اید (در کد اول) که شما از پارمتر اولی که در createServer() را فراخوانی کرده اید
نکته دیگر کد اول این است که ما فقط پارمترهای را به createServer() داده ایم و همان را نیز فراخوانده ایم . دلیل این موضوع این است که توابع جاوا اسکریپت می تواند توابع را مثل مقادیر به یک دیگر پاس دهند (برای اطلاعات بیشتر در خصوص رفتار و توابع در زبان های مختلف به کتاب "Concepts of Programming Languages 10th Edition" مراجعه کنید) .
برای مثال

```
function say(word) {
```

```
console.log(word);
}
function execute(someFunction, value) {
someFunction(value);
}
execute(say, "Hello")
```

به دقت به کد بالا نگاه کنید . اتفاقی که در کد بالا می افتد این است که تابع say را به صورت پارمتر به عنوان پارمتر اول تابع execute ارسال می شود .مقدار بازگشتی تابع say را به عنوان پارمتر تابع execute نشده است بکله خود تابع say به عنوان یک پارمتر ارسال شده است .

تابع say به متغییر محلی someFunction تابع execute پاس داده می شود (*اگر در خصوص پاس دادن مقادیر و متغییرها محلی و توابع آشنای کافی ندارید به کتاب شماره یک در قسمت منابع مراجعه کنید*) و تابع execute می تواند تابع موجود در متغییر somefunction() را صدا بزند . (در جهت تعریف یک تابع به عنوان یک متغییر ورودی تابع دیگر حتمی پرانتز گذاشته شود که تابع ی که در حال تعریف آن هستید متوجه شود که مقدار متغییر دریافتی از جنس تابع می باشد)

البته مشخص است که تابع say به دلیل اینکه یک پارمتر بیشتر ندارد , execute می تواند یک متغییر را زمان صدا زدن somefunction دریافت کنید .

می توان , همانطور که مشاهده کردید یک تابع یک پارمتری را به تابع دیگری بوسیله نام آن تابع پاس دهید و این توضیح به این شکل صادق است که می توان یک تابع را بصورت پارمتری در تابع دیگر تعریف کرد بصورت مثال :

```
function execute(someFunction, value) {
someFunction(value);
}
execute(function(word){ console.log(word) }, "Hello");
```

در کد فوق یک تابع را به عنوان تابع پارمتری دیگری تعریف کرده ایم .این یکی از ویژگی های جالب است که جاوا اسکریپت را متمایز از دیگر زبان ها .

چگونه تابع سرویس https را پاس می دهد

با توجه به اطلاعات کسب شده به عقب برگردیم به سراغ سرویس http :

```
var http = require("http");
http.createServer(function(request, response) {
response.writeHead(200, {"Content-Type":
"text/plain"});
response.write("Hello World");
response.end();
}).listen(8888);
```

اکنون در کد بالا مشخص است که چه اتفاقی برای createServer افتاده است .

حالا بررسی می کنیم که node.js چگونه کد ما ربصورت run time اجرا می کند روش اجرا run time روش مختصص node.js نمی باشد اما با زبان های دیگر مثل Python , Ruby , PHP یا جاوا متفاوت است

کد ساده زیر را مشاهده کنید :

```
var result = database.query("SELECT * FROM hugetable");
console.log("Hello World");
```

در حال حاضر قصد نداریم درباره اتصال به دیتا بیس صحبت کنیم بلکه فقط کد فوق جهت یک مثال ساده در بالا قرار گرفته است .

در اول مرحله اول کوئری از دیتا بیس ردیف های زیادی از دیتا بیس بر می گرداند و در خط دوم عبارت "Hello world" در کنسول قرار می گیرد. خواندن این همه ردیف از دیتا بیس (منظور دیتا بیسی با چندین میلیون رکورد می باشد) بسیار ترسناک است و چندین ثانیه طول می کشد .

به این روشی که کد را نوشتیم در جاوا اسکریپت مترجم node.js اول تمام مقدار برگشتی از دیتا بیس را می خواند و سپس می تواند تابع console.log() را اجرا کند .

اگر تکه کد بالا از طریق زبان php نوشته شده بود به روش یاد شده بالا کار می کرد. اول همه نتایج را می خواند و بعد خط بعدی از کد را اجرا می کرد و اگر این کد یک اسکریپت از صفحه وب بود کاربر باید چندین دقیقه برای لود شدن صفحه صبر می کرد .

به هر حال در روش مدل php این به یک مشکل بزرگ تبدیل نمی شود و وب سرور شروع به درخواست پروسس های php از هر درخواست دریافتی از httpمی کند . اگر یکی از نتایج درخواست های ارسالی در حال اجرا یکی از کد های مشابه بالا باشد خروجی این درخواست برای کاربر منحصر می باشد اما در خواست های کاربران دیگر برای صفحات دیگر را تحت تاثیر قرار نمی دهد .

نحوه اجرا این مدل در node.js متفاوت است در آنجا فقط یک پروسه وجود دارد و اگر یک درخواست به کد بالا داشته باشیم که باعث گرفتن کوئری با سرعت کم می شود در قسمتی از فرآیند این موضوع تمام پروسه های دریافتی را تا تمام شدن این کوئری تحت تاثیر قرار می دهد.

ما می دانیم با آنالیز این محتوا و دوباره نوشتن این کد مشکل حل خواهد شد به کد زیر توجه کنید

```
database.query("SELECT * FROM hugetable",
function(rows) {
var result = rows;
});
console.log("Hello World");
```

همانطور که در کد بالا مشاهده می کنید database.query() مقدار بازگشتی را به پارمتر دوم که بصورت تابع تعریف شده است پاس می دهد .

در حالت قبلی این کد اول کد کوئری دیتابیس را اجرا می کرد و بعد از خاتمه کنسول نوشتن اجرا می شد. حالا node.js می تواند با هندل درخواست ناهنگام از دیتا بیس

در متن بعدی مفصلا در خصوص دیتابیس و نودجی اس بحث خواهیم کرد

Node.js و express

اجرا کردن پروژه با express

برای اجرا کردن تمامی پروژه های که با Express می سازیم رویه زیرا دنبال کنید :

۱- فایلی با نام package.json [۴] با محتوای زیر در شاخه ای مورد نظر بسازید (بصورت مثال cd express-app $ در ریشه مورد علاقه)

```
{
"name": "test-app",
"version": "0.0.1",
"private": true,
"scripts": {
"start": "node app"
},
"dependencies": {
"express": "3.2.6",
"jade": "*"
}
}
```

[4] JavaScript Object Notation

توضیح فیلد های استفاده شده در بالا به شرح زیر می باشد

فیلد	توضیح
Name	نام انتخابی شما برای ماژولی که قصد ساختن ان را دارید
Version	نسخه ماژول را از اینجا معرفی می کنید
Private	اجازه پابلیک بودن یا نبودن آن در ریجستری npm از این طریق تنظیم می شود که در این پروژه مقدار true دارد که تعیین می کند اپلکیشن بصورت شخصی (خصوصی) می باشد
Scripts	دستورات اجرای npm در پروژه بالا بصورت ویژه دستور node app فقط دستور اجرای ماژول می باشد .
Dependencies	ماژول های وابسته در اینجا معرفی می کنید که در بالا ماژول های express و jade بصورت ویژه معرفی شده اند.

بعد ساختن فایل pakage.json در هر شاخه ی که فایل مورد نظر ساخته شده است دستور زیر را در CMD اجرا نمایید

C:\ express-app\ npm install

یک اپ ساده با express

خوب اپ ساده زیر با با نام app.js در شاخه ی که در بالا ساختید با محتوای زیر ذخیره کنید و دستور node app را در همان شاخه در CMD اجرا نمایید .

```
// کتابخانه http
var http = require('http');
// ماژول express
var express = require('express');
```

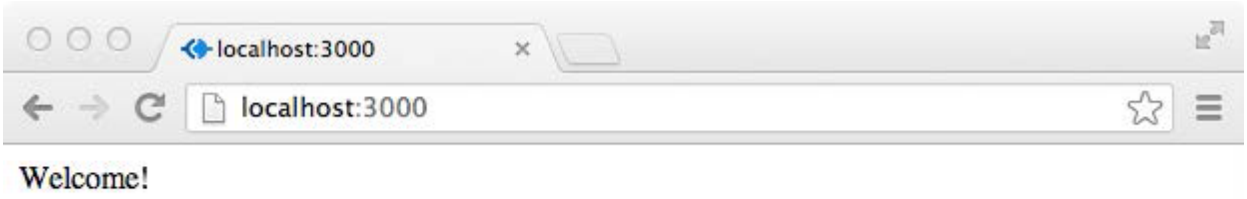
```
// یک نمونه ساخته شده از express
var app = express();
// شروع اپ
http.createServer(app).listen(3000, function() {
console.log('Express app started');
});
// یک روت برای صحفه خانه
app.get('/', function(req, res) {
res.send('Welcome!');
});
```

برای اجرا اپ بالا دستور node app را اجرا کنید . بعد از اجرا پیام "**Express app started**" مشاهده خواهید کرد و برای پایان دادن به اجرا سرور دکمه ctrl+c را کلیک کنید .

نکته :در مورد پیام خطا 404 و ۵۰۰ و هندل های آن در ادامه توضیح خواهد داده می شود

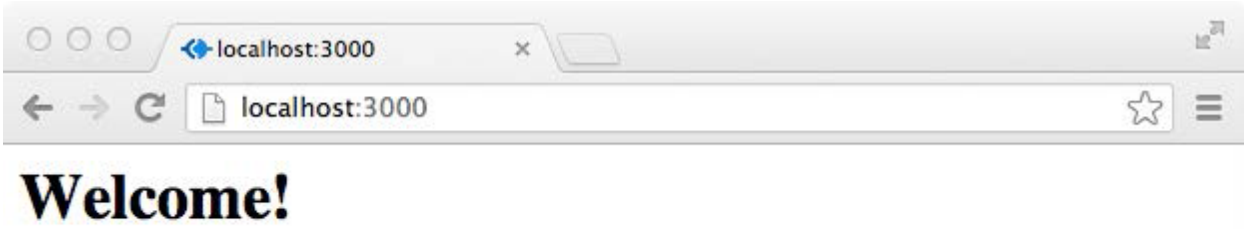
آنالیز خروجی:

کلمه welcome در آدرس http://localhost:3000 در صفحه اصلی نمایش داده می شود و در دیگر در خواست های مقدار پیام خطا ۴۰۴ را بر می گرداند.



اگر سورس صفحه را نگاه کنید می بنید که یک پاسخ بصورت تکس دریافت کرده اید . سوالی که الان مطرح است برای پاسخ بصورت html چه باید کرد !؟

خوب بصورت نمونه خط res.send('Welcome!') مقدار متنی را به <h1>Welcome!</h1> تغییر دهید و سرور رو ری استارت کنید و صحفه را در مرورگر رفرش کنید و خروجی مانند تصویر زیرا را خواهید دید .



نکته : برای دیدن خروجی جدید حتمی با سرور را restart کنید و یا اینکه از supervisor استفاده کنید با کمک این ماژول بعد از هرگونه تغییر سرور را بصورت اتوماتیک restart می کنید برای اطلاعات بیشتر درباره supervisor و آموزش استفاده به اینجا https://github.com/isaacs/node-supervisor مراجعه کنید.

حالا چگونگی ارسال یک صفحه html را دنبال خواهیم کرد . اپ های Express کامپونت ویژه ای دارند به نام views دارد. هرجا که شما لازم داشته باشید از html بصورت زبان قالب استفاده کنید هر تغییری در فرم html هسته app نیاز باشد بوسیله views اعمال و بصورت html در خروجی بدون نیاز به restart کردن سرور تغییر و ذخیره خواهد کرد.

express و نمایش :

با محوریت نمایش محتوا اپی می نویسم . خوب برای این منظور شاخه ای باید ایجاد شود . همیشه این موضوع را در نظر بگیرد نام گذاری ها به شکلی باشد که با محتوای آن در ارتباط باشد این موضوع ساده به بهتره و قابل درک بودن اپ شما کمک می کند.

شاخه ای با نام views در مسیر مورد نظر بسازید یا با دستور **mkdir views** در ترمینال سیستم عامل خود.

حالا در شاخه views دو فایل با نام های index.jade برای نمایش صفحه خانه و hello.jade برای صفحه سلام بسازید . اگر شما یک برنامه نویس باشید حتمی الان کنجکاو شده باشید چرا پسوند JADE و یا اصلن این پسوند چیست ؟

فایل index.jade در مسیر views بسازید و محتوای زیر را در درون آن قرار دهید

```
doctype html
html(lang="fa" dir="rtl")
  head
    title= page

  body
    h1 جی-د سازنده قالب

    h3 شما شگفت انگیز هستی
    p پیش به سوی آینده
    p.
                جی-د یک زبان سازنده قالب که در ساخت آن کارایی و سادگی در نظر گرفته شده است
```

فایل hello.jade در مسیر views بسازید و محتوای زیر را در درون آن قرار دهید

```
html(lang="fa" dir="rtl")
head
title Hello
body
b Hello!
p سلام
```

JADE چیست

JADE در معنی لغوی اسم نوع سنگ قیمتی و زینتی به نام یشم سبز می باشد اما JADE یا معادل پینگلیش آن جی-د در اینجا زبانی است با ساختار شی گرا [5] که اولین بار در مرکز نیوزلند بنیان گذاری شد و اولین نسخه آن در سال ۱۹۹۶ ساخت شد .

براساس ساختارend-to-end طراحی شده است و همچنین شامل API برای زبان های دیگر نیز می شود شامل Net Framework. و JAVA و C/C++ و وب سروریس ها .

JADE و HTML

جی-د از ساختارهای متفاوتی پشتیبانی می کند اما برای دور نشدن از مبحث بصورت ویژه در خصوص پشتیبانی آن از HTML توضیح می دهیم.

جی-د خیلی ساده تر از ASP.NET در خصوص خلق و توسعه قالب های HTML و صفحات کار می کند و زمانی که یک قالب برای صفحه HTML طراحی می شود بصورت چند بخشی طراحی می شود و بر خلاف ساختار کلی جی-د در زمان کار با ساختار HTML از متد Front End استفاده می کند .

[5] Object-oriented

NODE.JS و JADE

جی-د در نود یک موتور ساخت قالب می باشد و رندر پیش فرض در فریم ورک express می باشد و به همین دلیل از این زبان در طول این متن استفاده خواهد شد و در صورت امکان از متدهای دیگر نیز مثالی زده خواهد شد .

برای آشنای بیشتر با این ساختار به منابع زیر می توانید مراجعه کنید :

۱- http://jade-lang.com/
۲- www.franz-enzenhofer.com
۳- تبدیل html به jade : html2jade.aaron-powell.com

ساخت یک اپ ساده با express

برای شروع یک شاخه جدید در مسیر مورد نظر خود بسازید که در اینجا ما در مسیر express-app شاخه ۱ را ساخت ایم و فایل به نام app.js که محتوای زیر در درون آن قرار گرفته است(فایل های این اپ بصورت پیوست قرار دارد)

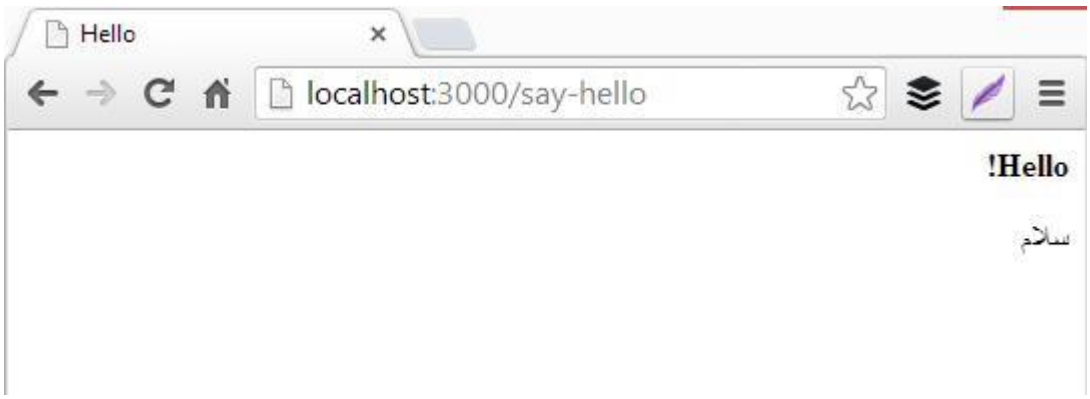
اپ شماره ۱ :

```
var http = require('http');
var express = require('express');
var app = express();
// موتور نمایش قالب را مشخص می کند
app.set('view engine', 'jade');
// مسیر نمایش را مشخص میکند
app.set('views', './views');
//    صفحه اصلی با کمک تابع زیر تنظیم می شودصفحه
// index
// از این طریق به عنوان صفحه اصلی انتخاب می شود
```

```
app.get('/', function(req, res) {
res.render('index');
});
// masire say0hello az in tarigh be view render
mishavad
app.get('/say-hello', function(req, res) {
res.render('hello');
});
//  از این طریق زمانی که مسیر تست درخواست شود مقدار متنی
// this is a test
//  نمایش داده می شود
app.get('/test', function(req, res) {
res.send('this is a test');
});
//  مانند تابع بالا عمل می کند زمانی که مسیر
//http://127.0.0.1:3000/jade
//  درخواست می شود مقدار اچ تی ام ال نمایش داده می شود
app.get('/jade', function(req, res) {
res.send("<h3>Jade is a terse and simple templating
language .</h3>");
});
http.createServer(app).listen(3000, function() {
console.log('App started');
});
```

اپ از طریق پرت ۳۰۰ اجرا و منتظر پاسخ و دریافت اطلاعات می ماند در اینجا بهتر است بگویم دریافت مسیر جدید خواهد بود .در زمان درخواست مسیر اصلی به آدرس : http://127.0.0.1:3000/ محتوا صفحه Index.jade را نمایش می دهد (تصویر شماره ۱) و در هنگام در خواست مسیر http://127.0.0.1:3000/say-hello صفحه hello.jade به نمایش در می آید (تصویر شماره ۲) و همچین موقع درخواست از مسیر http://127.0.0.1:3000/jade و http://127.0.0.1:3000/test محتوای که به آن در تابع مقدار دهی کردیم نمایش داده خواهد شد (تصویر شماره ۳ و ۴).

تصویر شماره ۱



تصویر شماره ۲

تصویر شماره ۳

تصویر شماره ۴

نکته : اگر سورس هر کدام از صفحه ها را نگاه کنید می بنید نمایش بصورت خروجی HTML انجام شده است و در صورت تغییر در هر کدام از صفحات jade نیاز به restart کردن اپ نمی باشد مگر اینکه در ساختار app.js تغییر ایجاد کنید.

یک دایرکتوری عمومی برای اپ

هم اکنون چند قدم به داشتن توانای و دانش ساخت یک وب سایت تابعی و داینامیک نزدیکتر شدیم . حال به محتوا می خواهیم CSS , JS[۶] و تصاویر اضافه کنید و محل ذخیره سازی این فایل ها یک دایرکتوری عمومی خواهد بود .

Express از متد استاتیک middleware[۷] برای صدا زدن استفاده می کند

[6] JavaScrip

از روش زیر می توانید یک دایرکتوری را بعنوان منبع استاتیک تنظیم ویا به عبارت بهتر اضافه کنید

```
app.use(express.static('./public'));
```

و برای اضافه کردن چند دایرکتوری استاتیک در صورت نیاز می توانید بصورت زیر استفاده کنید

```
app.use(express.static('./public'));
app.use(express.static('./files'));
app.use(express.static('./downloads'));
```

حالا روش ساخت پروژه جدید را انجام دهید و بعد از آن دایرکتوری های زیر را بسازید

$ mkdir public
$ mkdir public/images

$ mkdir public/javascripts
$ mkdir public/stylesheets

این اسامی دایرکتوری جهت واضح و قابل درک بودن اپ انتخاب شده اند و شما می توانید در اپ های دیگر اسامی دلخواه خود را انتخاب کنید اما سعی کنید اسامی انتخاب کنید باعث پیچیدگی درک اپ برای خود و یا دیگر برنامه نویس ها نشود .

خوب یک تصویر به عنوان لوگو[8] با نام logo.png در دایرکتوری images ازاضافه کنید من از لوگو وب برو استفاده کردم

یک فایل به نام main.js بسازید در دایرکتوری javascripts با محتوا زیر بسازید.

```
window.onload = function() {
document.getElementById('smile').innerHTML = ':)';
};
```

یک فایل دیگر به نام style.css در دایرکتوری stylesheets با محتوا زیر بسازید:

[7] میان‌افزار
[8] Logo= لوگو = آرم

```
#content {
	width: 220px;
	margin: 0 auto;
	text-align: center;
	border: 1px solid #ccc;
	box-shadow: 0 3px 4px #ccc;
	padding: 5px;
	font-family: Tahoma;
	font-weight: normal;
	color: black;
	font-size: 12px;
}
#smile
{
	-webkit-transform:rotate(-90deg);
	-moz-transform:rotate(-90deg);
	-o-transform: rotate(-90deg);
	font-family: sans-serif;
	font-size: 18px;
	font-weight: normal;
	color: orange;
}
```

فایل index.jade را به روز کنید

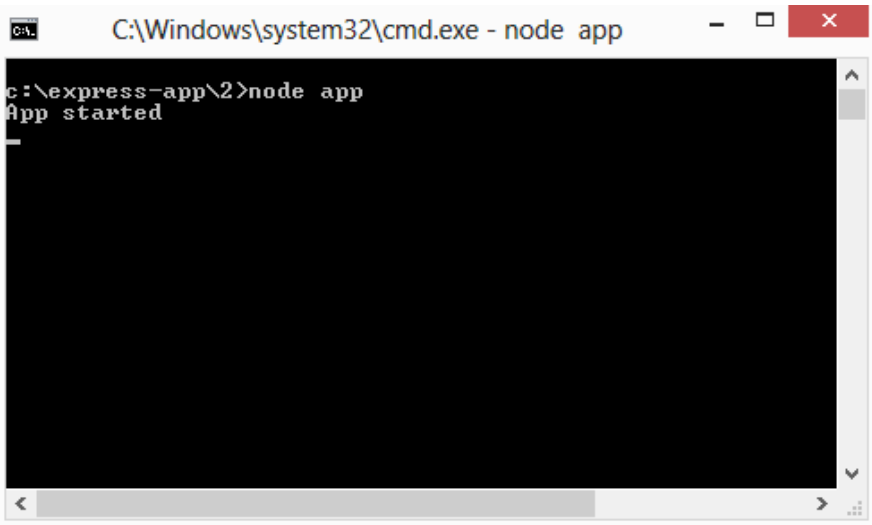
```
doctype html
html(lang="fa" dir="rtl")
title Welcome
script(src='javascripts/main.js')
link(rel='stylesheet', href='stylesheets/style.css')
body
#content
	img(src='images/logo.png')
	p خوش آمدید
	#smile
```

و کد زیر را در فایل app.js ذخیره کنید

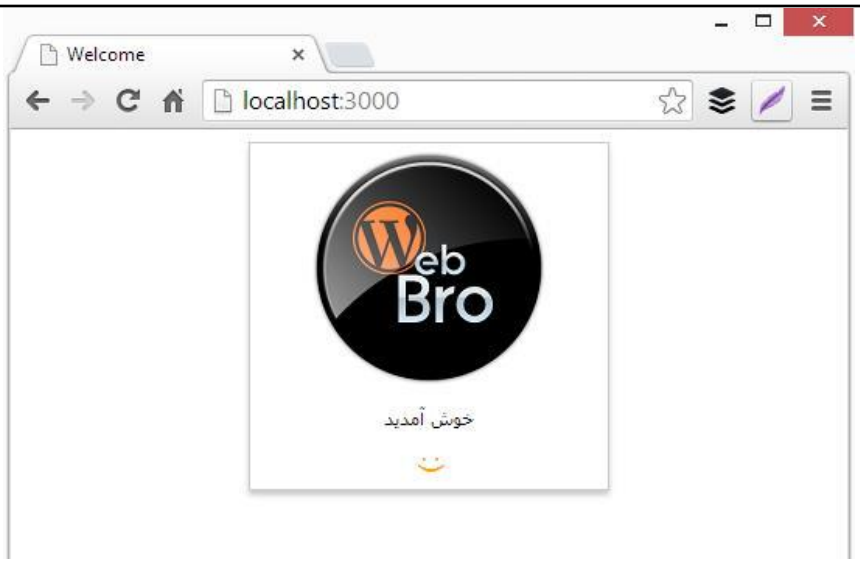
```
var http = require('http');
var express = require('express');
var app = express();
```

```
app.set('view engine', 'jade');
// دایرکتوری ها از طریق مشخص می شوند
app.set('views', './views');
// دایرکتوری
//public
// بصورت استاتیک از طریق زیر
//برای دسترسی به اپ معرفی شده است
app.use(express.static('./public'));
// صفحه اصلی مشخص شده است
app.get('/', function(req, res) {
res.render('index');
});
// سرور لوکال ساخته روی پورت ۳۰۰۰ منتظر دستور
http.createServer(app).listen(3000, function() {
console.log('App started');
});
```

اپ را اجرا کنید با دستور node : مانند تصویر زیر :



خوب خروجی مثل همیشه روی آدرس http://localhost:3000 در دسترس می باشد مانند تصویر زیر



تبریک می گم اولین اپ با express خود را ساختید .

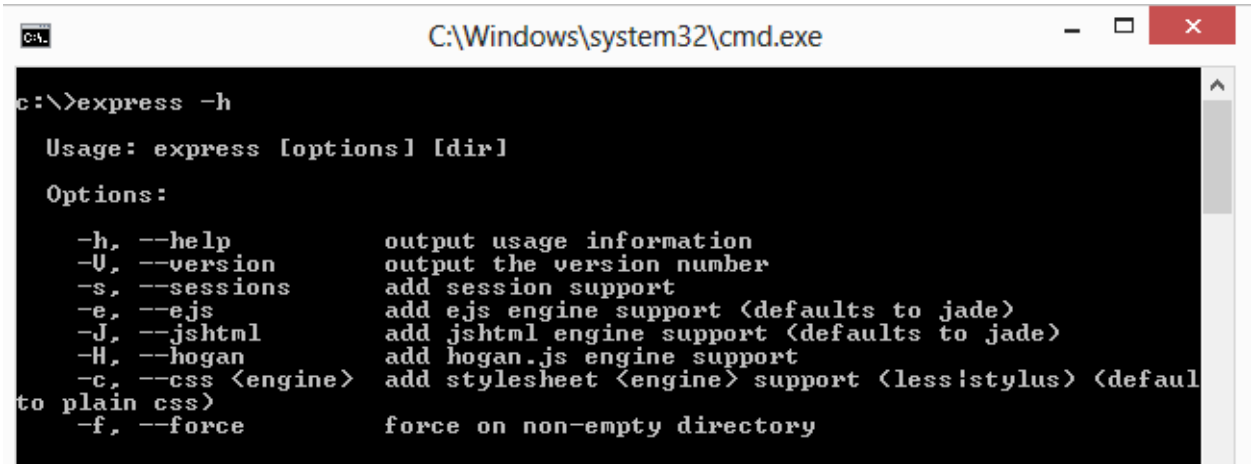
اپ شماره ۲

مولد خودکار app در express

Express یک دستور خودکار جهت تولید اپ و دایرکتوری های views و ... غیر را دارد . جهت استفاده از این دستور و دیگر دستورات در ترمینال سیستم عامل خود تایپ کنید

- **express –h**

تا مانند تصویر زیر این اطلاعات نمایش داده شود .

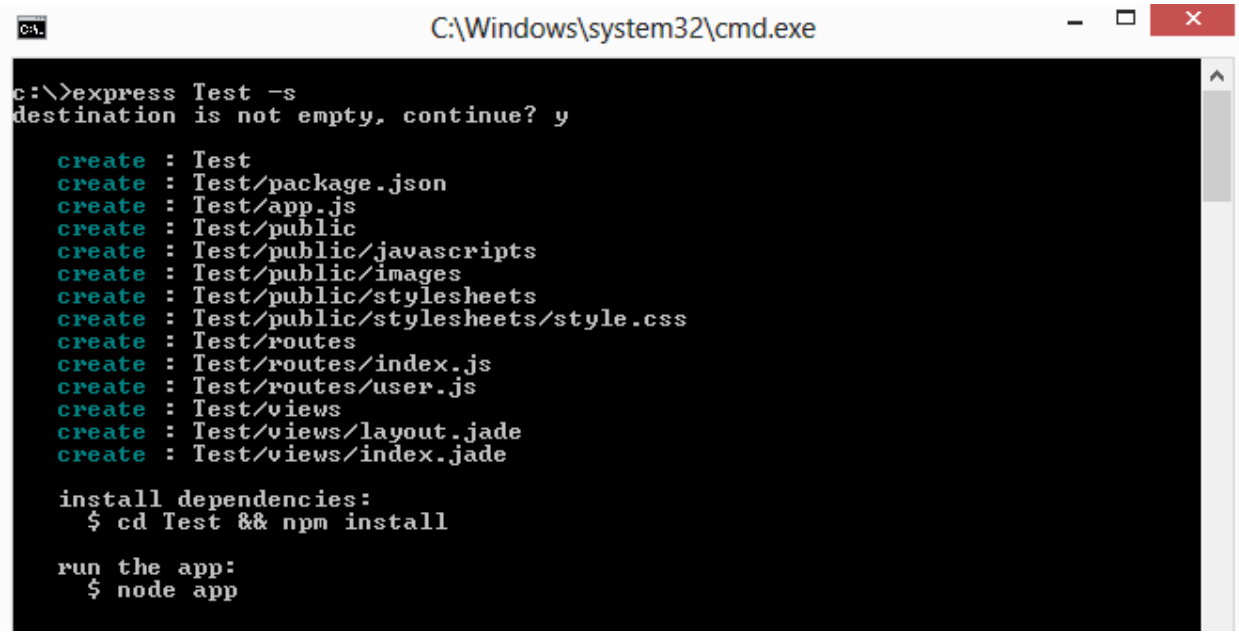


همانطور که در تصویر فوق می بینید روش استفاده توضیح داده شده است

روش کلی استفاده به این شرح است که اول دستور express و بعد فرمان انتخابی شما و نهایتن مسیر یا دایرکت که قرار است فرمان روی آن اعمال شود را وارد می کنید .

فرمان کمک ونمایش همه فرمان ها	**-h**
نمایش نسخه express	**-V**
ساخت یک اپ	**-s**
پستیبانی از موتور جی-د	**-e**
پشتیبانی از موتور jshtml (پیش فرض)	**-J**
اضافه کردن موتور hogan.js برای پشتیبانی (پیش فرض)	**-H**
اضافه کردن css (پیش فرض)	**-c**
نمایش محتوا در مسیر داده شده	**-f**

بصورت نمونه روش استفاده از فرمان express در تصویر زیر نمایش داده شده است



حالا آپ ساخته شده جدید را اجرا کنید و خروجی را مشاهده کنید به کمک دستور همیشگی :

- **node app**

همانطور که در تصویر فوق مشخص است تمام ساختار شبیه آنچه تا به الان می ساختیم بجز دایرکتوری routes و فایل های درون آن عجله نکنید به زودی درباره این دایرکتوری و فایل های موجود در آن نیز صحبت خواهیم کرد.

دیگر نیازی نیست برای هر پروژه فایل ها و دایرکتوری ها را ایجاد کنید با کمک خط فرمانی تمامی آنچه برای یک اپ نیاز دارید خلق می کنید .

نکته ی که درباره این مولد باید در نظر بگیرد تمام این محتوا تولید بصورت پیشنهاد می باشد و شما به دلخواه هر تغییری که بخواهید می توانید در آن ایجاد کنید .

قدرت دادن express با خاصیت میان افزاری

به خاطر دارید که app.use() چگونه به عنوان یک میان افزار استفاده می کردیم . حالا به این موضوع فکر کنید که ما می توانیم میان افزار خود را کد نویسی کنیم ☺ حالا به استفاده میان افزارهای که به همراه expressJs است توجه کنید

میان افزار	توضیح
router	مسیر سیستمی اپ
logger	درخواست لوگ به سرور
compress	پشتیبانی gzip/deflate در سرور
basicAuth	تصدیق HTTP
json	پارس [9] اپلکیشن json
urlencoded	پارس اپلکیشن از urlencoded [10]
multipart	پارس چند بخشی / فرم دیتا
bodyParser	پارس کردن متن درخواست / در قالب JSON یا urlencoded و میان افزار چند بخشی
timeout	پایان زمانی درخواست

[9] Parse به عمل تجزیه و تحلیل کد توسط کامپایلر یا خواندن آن توسط انسان گفته می شود
[10] Urlencoded : فراخوانی رمزنگاری(منظور متد ویا زبان مورد استفاده) از مسیر تعریف شده شده را دارد

cookieParser	پارس کننده [11] کوکی
session	پشتیبانی از Session
cookieSession	کوکی بر پایه سشن
methodOverride	پشتیبانی از متد http
responseTime	نمایش مدت زمان پاسخگوی سرور
static	دایرکتوری ثابت و باارزش و اهمیت برای یک وب سایت
staticCache	نهان[12] کردن برای میان افزار ثابت
directory	مسیر دایرکتوری ها
vhost	فعال کردن میزبانی مجازی [13]
favicon	Favicon برای وب سایت
limit	محدود کردن سایز متن درخواندن
query	پارسر کردن کوئری GET
errorHandler	تولید یک پشته با فرمت HTML و دنبالی از پیام خطاهای سرور

در بالا لیستی از میان افزار های پیش فرض که در دسترس قرار دارد موجود می باشد و در زیر برای مثال از جگونگی به کار بستن یک میان افزار که در زیر از responseTime استفاده شده است را مشاهد می کنید

فایل app.js را به محتوای زیر جهت استفاده از میان افزار تغییر دهید

```
var http = require('http');
var express = require('express');
var app = express();
app.set('view engine', 'jade');
app.set('views', './views');
app.use(express.static('./public'));
// زمان پاسخ گوی را اضافه می کند
app.use(express.responseTime());
app.get('/', function(req, res) {
res.render('index');
[ 41 ]
});
```

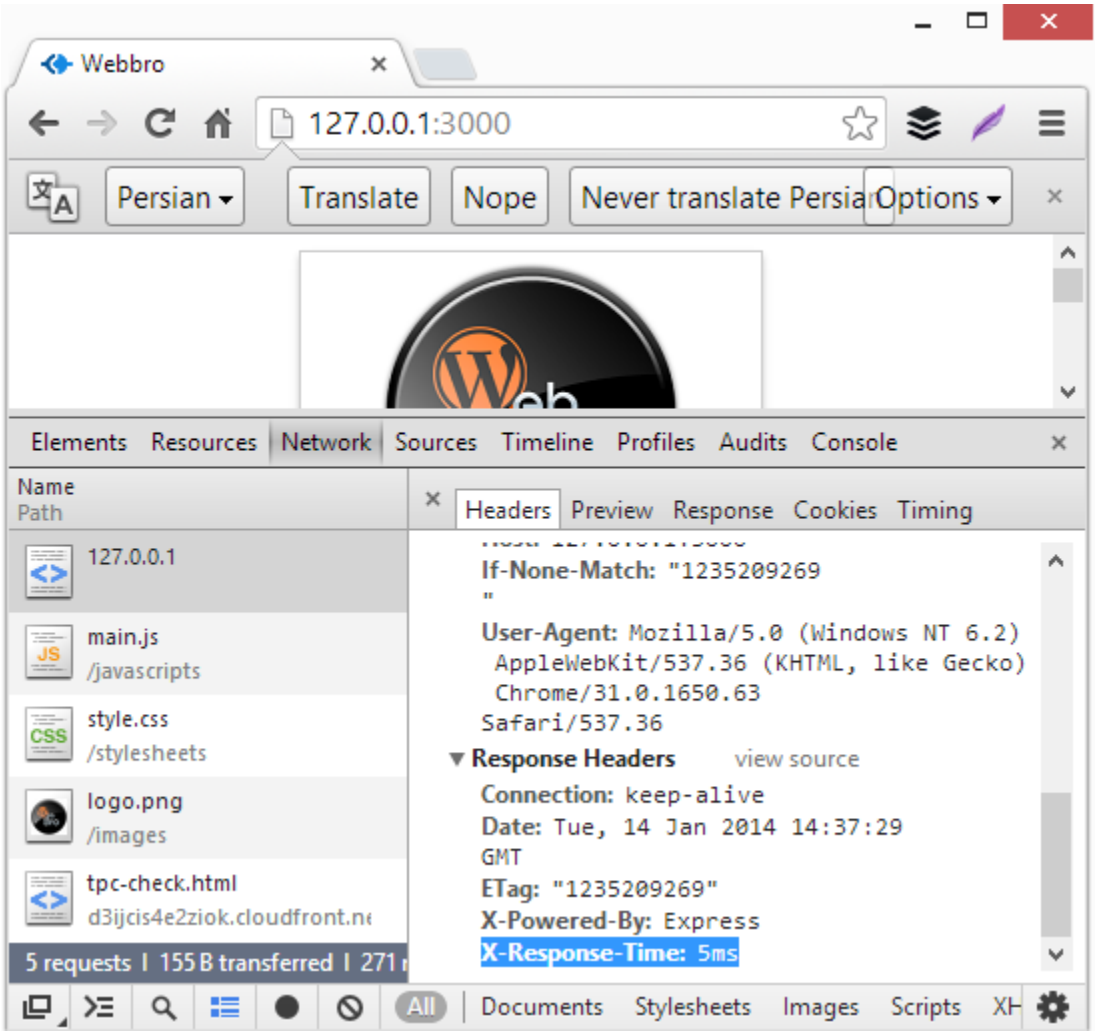
[11] Parser
[12] Cache
[13] Vhost

```
http.createServer(app).listen(3000, function() {
console.log('App started');
});
```

اپ شماره ۳

خوب مثل همیشه با دستور : node app اپ را اجرا کنید و بعد از اجرا خروجی تصویر زیر را مشاهده می کنید .

همانطور که در تصویر مشخص است از مرورگر کروم استفاده کرده ام که زمان پاسخگوی http به مرورگر در تب شبکه ابزار توسعه [۱۴] نمایش داده شده است .



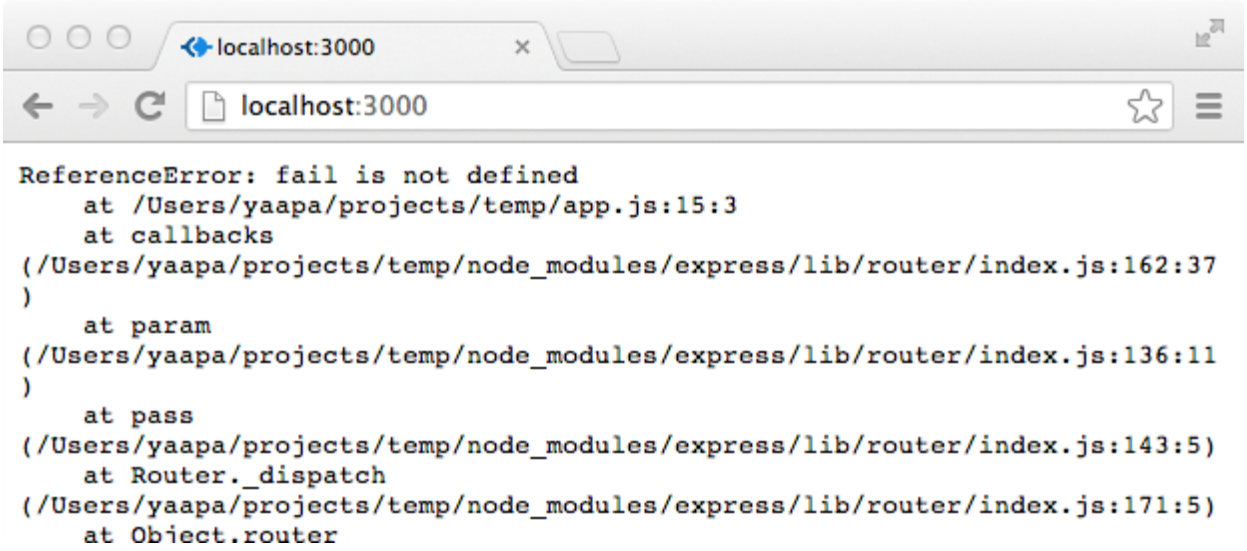
[14] Chrome developer tool

در مرحله بعدی می خواهیم میان افزار handle error را بررسی کنیم . برای بررسی این handler یک خطا تولید می کنیم و این عمل را با صدا زدن یک تابع تعریف نشده انجام می دهیم و بر طبق errorHandler توضیحات چگونگی این Error به صورت HTML نمایش داده خواهد شد به کد زیر توجه کنید .

```
var http = require('http');
var express = require('express');
var app = express();
app.set('view engine', 'jade');
app.set('views', './views');
app.use(express.static('./public'));
app.use(express.responseTime());
// اضافه کردن میان افزار هندلر ارور
app.use(express.errorHandler());
app.get('/', function(req, res) {
// تابع که تعریف نشده است جهت تولید خطا صدا می زنیم
fail();
});
http.createServer(app).listen(3000, function() {
console.log('App started');
});
```

کد بالا مانند به بقیه کد های app.js پیشین می باشد با این تفاوت که تابع fail(); را فراخوانی کرده ایم بدون اینکه آن را تعریف کنیم جهت تولید خطا و دیدن چگونگی کارکرد errorHandler() که در این اپ همانطور که در کد بالا می بنید به اپ اضافه شده است .

مثل همیشه برای اجرای app از دستور node app در شاخه مورد نظر استفاده می کنید (پروژه شماره ۴) و خروجی آن را در تصویر زیر می توانید مشاهده کنید



خروجی تصویر فوق مانند یک صفحه HTML نمی باشد در حقیقت شما می توانید تنظیم کنید که بصورت HTML نمایش داده شود. اما سوال چرا میان افزار errorHandler کار نکرد ؟

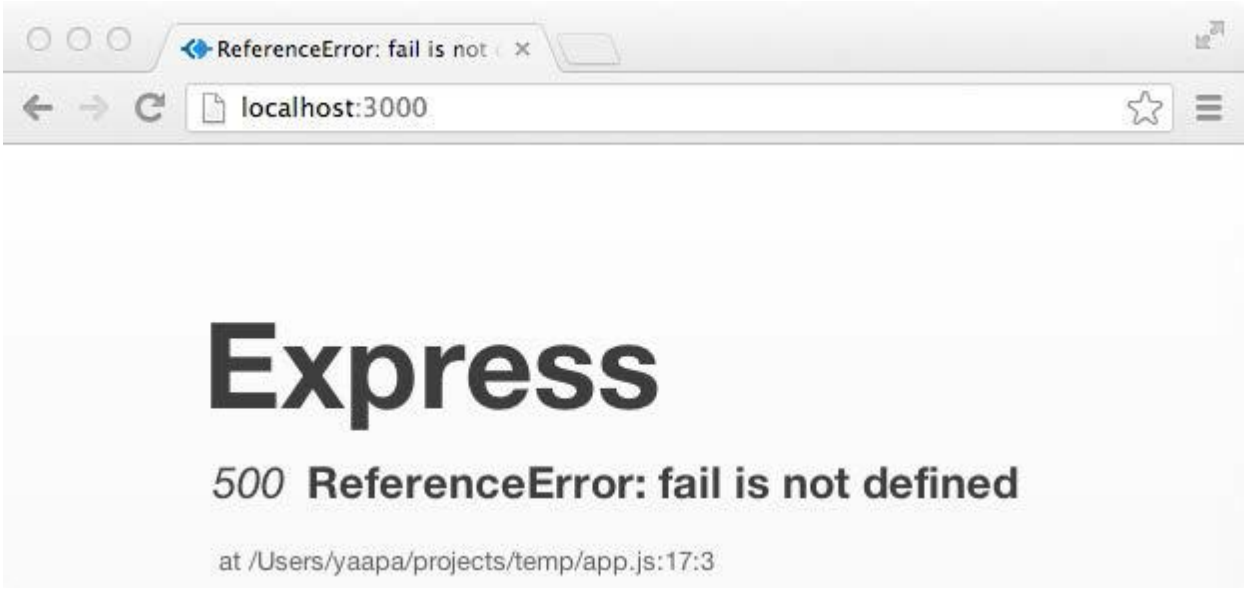
همانطور که در اول این مبحث اشاره شد بودن اضافه کردن میان افزار router بصورت مشخصی اپ مسیر تعریف شده را نمایش می دهد . مهمترین درخواست در میان افزار errorHandler این می باشد که پیش از آن باید میان افزار router را اضافه کنید . هیجان زده نشوید آنگونه که باید کار کند نمی شود .

طبق انتظارات و بدیهیات میان افزار router را جهت تولید و استفاده از errorHandler بکار می بندید کد زیر را مشاهده کنید (پروژه شماره ۵)

```
var http = require('http');
var express = require('express');
var app = express();
app.set('view engine', 'jade');
app.set('views', './views');
app.use(express.static('./public'));
app.use(express.responseTime());
//  اضافه کردن میان افزار روتر
app.use(app.router);
//  اضافه کردن میان افزار اورهندلر
app.use(express.errorHandler());
app.get('/', function(req, res) {
//  صدا زدن تابع تعریف نشده جهت تولید خطا
```

```
fail();
});
http.createServer(app).listen(3000, function() {
console.log('App started');
});
```

خوب کد بالا را اجرا کنید و خروجی مانند تصویر زیر مشاهده خواهید کرد



این بار errorHandler خروجی نمایش داد که انتظار آن می رفت در خروجی که بصورت HTML می باشد اعلام می کند تابع fail تعریف نشده است و پیام خطا مبدا با شماره ۵۰۰ را می دهد

شما می توانید ازerrorHandler هر زمان که نیاز داشتید از آن استفاده کنید . Express به صورت پیش فرض کار زیادی انجام نمی دهد در هنگام دی باگ اپ شما ولی این میان افزار بسیار مفید خواهد بود

اخیرا دیدم که شما میت وانید میان افزار خود را بنویسد البته اگر بخواهید . هر کاری که شما بخواهید انجام دهید با دو آبجکت (شی) [15] req و res عملی می شود

[15] Object

Express قدرت گرفته از ماژول Node

Express یک بسته با کتابخانه ی عظیم برای راه اندازی وب سایت نیست اما این به امر به این معنی نیست که قابلیت تولید تسک [16] (وظایف) را ندارد

شما مجموعه ی عظیم از ماژول های نود را در ریجستری npm بصورت پلاگین در دسترس دارید و به آسانی می توانید از ان در اپ های خود استفاده کنید.

در فصل قبل توضیح دادیم ماژول نود را چگونه تولید و بنویسیم . ما از آن طریق می توانیم به نوشتن وسیع و قدرت مند در Express بپردازیم

شما با ماژول های نود می توانید خیلی چیزها را بنویسید وبه سرانجام برسانید اما هر چیزی که سعی دارید بنویسید بصورت متن باز[17] ماژول نود آماده استفاده می باشد و فقط احتیاج است شما ماژول درست را پیدا و نصب کنید و از آن در اپ خود استفاده کنید.

نکته npm بصورت عمومی در دسترسی همه توسعه دهندگاه نود جهت انتشار و استفاده ماژول های نود موجود می باشد .ماژول های موجود با دستور npm نصب می شوند

لیست عظیم و بزرگی از ماژول های موجود در این آدرس موجود می باشد
https://github.com/joyent/node/wiki/

چگونگی روش نصب NPM بوسیله یک فایل ini. بصورت مثال یک ماژول پارسر به نام iniparser را نصب و در اپ که می سازیم استفاده خواهیم کرد .

```
$ npm install iniparser
npm WARN package.json application-name@0.0.1 No README.md file found!
npm http GET https://registry.npmjs.org/iniparser
npm http 304 https://registry.npmjs.org/iniparser
iniparser@1.0.5 node_modules/iniparser
```

[16] Task
[17] Open Source : OS

همانطور که در تکه کد بالا می بنید با دستور npm install iniparser در ترمینال سیستم عامل این ماژول نصب می شود .

حالا در ادامه ساخت اپ یک فایل به نام config.ini با محتوای زیر در مسیر اصلی اپ بسازید

```
title = اپ وب برو

port = 3000

message = شما عالی هستید !
```

هنوز کار ما تمام نشده است حالا باید فایل app.js را بصورت زیر تغییر دهید .

```
var http = require('http');
var express = require('express');
var app = express();
// ماژول پارسر را اضافه میکند
var iniparser = require('iniparser');
// محتوای فایل کانفیگ را می خواند
var config = iniparser.parseSync('./config.ini');
// قسمت های زیر پیشتر توضیح داده شده است
app.set('view engine', 'jade');
app.set('views', './views');
app.use(express.static('./public'));
app.use(express.responseTime());
app.use(app.router);
app.use(express.errorHandler());
app.get('/', function(req, res) {
// دو متغییر کانفیگ به ویور پاس داده می شود
res.render('index', {title:config.title,
message:config.message});
});
http.createServer(app).listen(config.port, function() {
console.log('App started on port ' + config.port);
});
```

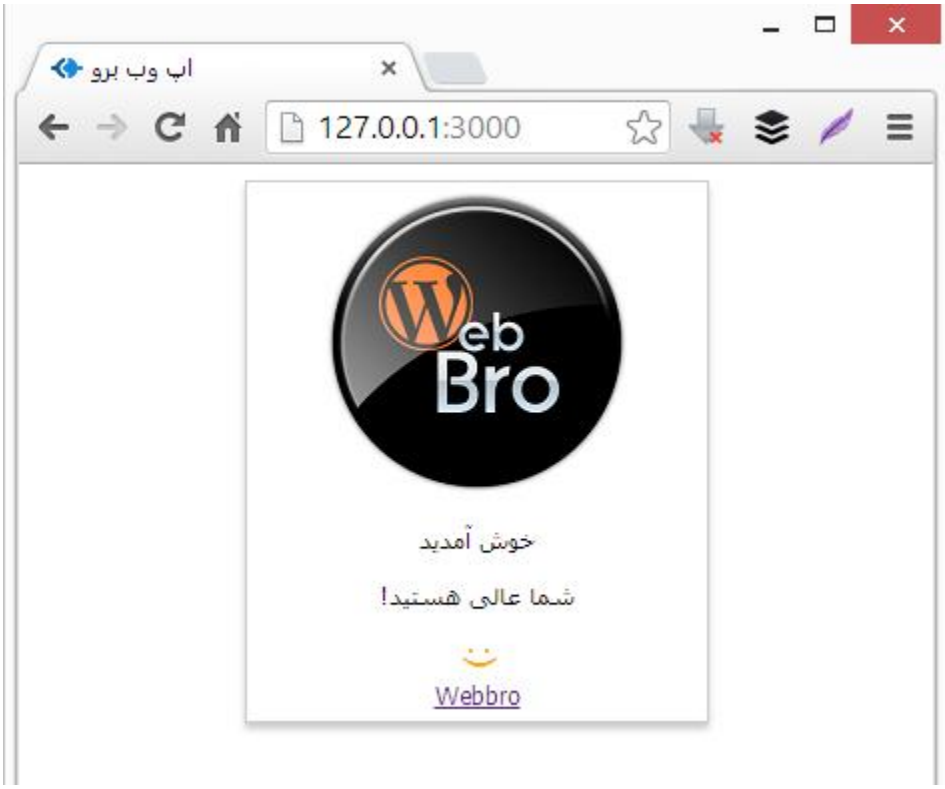
و فایل jade که وظیفه نمایش را دارد به شکل زیر تغییر می دهیم .

```
doctype html
html(lang="fa" dir="rtl")
title #{title}
script(src='javascripts/main.js')
link(rel='stylesheet', href='stylesheets/style.css')
body
#content

    img(src='images/logo.png')
    p آمدید خوش
    P #{message}
    #smile
    a(href='http://www.webbro.ir', target='_blank')
Webbro
```

پروژه شماره ۶

حالا وقت اجراست . در تصویر زیر اجرای خروجی اپ فوق قرار دارد .



همانطور که متوجه شده اید می توانید با کمک config.ini از زبان فارسی برای دادن عنوان و ... غیر نیز استفاده کنید .پارسر از تمام استاندارد Unicode پشتیبانی می کند .

Express بصورت ذاتی توانانی پارس کردن فایل ini. را ندارد اما می توان با توسعه آن از طریق ماژول iniparser این قابلیت را به express اضافه کرد همانطور که در اپ این اتفاق افتاد.

درخواست log به اپ

Express ماژولی دارد به نام logger که بسیار مفید برای ساخت و توسعه اپ ها می باشد . برای استفاده از این ماژول با تعریف خط زیر می توانید به راحتی آن را به اپ خود اضافه و از آن استفاده کنید

```
app.use(express.logger());
```

بدون هیچ تغییر ماژول log توضیحات ورودی را به همراه دارد . شما می توانید آن را از طریق نشانه های [18] موجود که در زیر قرار دارد بصورت سفارشی از آن استفاده کنید .

محتوا	نشانه
درخواست بصورت ویژه با HTTP	:req[header]
پاسخ بصورت ویژه با حالت HTTP	:res[header]
ورژن HTTP	:http-version
مدت زمان جهت تولید پاسخ	:response-time
آدرس IP کاربر	:remote-addr
تاریخ و زمان درخواست	:date
متد HTTP استفاده شده برای یک درخواست	:method
URL درخواست کننده	:url
URL مراجعه کننده URL جاری	:referrer
اطلاعات کاربر	:user-agent
وضعیت HTTP	:status

[18] Token

و در زیر چگونگی فرمت استفاده از نشانه های بالا قرار دارد

```
app.use(express.logger({ format: ':remote-addr :method
:url' }));
```

در زمان استفاده از خط بالا در اپ و بعد از مراجعه به صفحه در ترمینال خود پیام مشابه زیر را مشاهده خواهید کرد.

127.0.0.1 GET /
127.0.0.1 GET /favicon.ico

بصورت پیش فرض خروجی logger در ترمینال نمایش داده می شود . اما ما می توانیم با کمک گزینه stream وضعیت log را در یک فایل ذخیره کنیم . کد زیر را مشاهده کنید .

```
var http = require('http');
var express = require('express');
var fs = require('fs');
var app = express();
app.use(express.logger({
format: 'tiny',
// ذخیره سازی لاگر در فایل
stream: fs.createWriteStream('app.log', {'flags': 'w'})
}));
...
```

Logger از چهار حالت تعریفی با فرمت های Default , Short , Tiny و Dev پشتیبانی می کند و شما می توانید بصورت خاص هر کدام را نیز از طریق زیر تعریف و استفاده کنید . به کد زیر توجه کنید :

```
app.use(express.logger('dev'));
```

استفاده از یک فایل پیکربندی شده

بصورت عمومی نیاز به استفاده از فایل ini. برای پیکر بندی اپ های ما نمی باشد در مثال های بعدی متوجه چگونگی این موضوع خواهید شد . در مثال های که تاکنون دیدم فقط استفاده از ماژول نود را نشان دادیم نه توصیه های عملی .

حالا از زاویه ای چگونگی کار کرد require() را خواهیم دید. نود از پیکربندی فایل JSON-based بصورت پیش فرض پشتیبانی می کند . ایجاد یک فایل بوسیله یک آبجکت (شی) JSON و عمل ذخیر سازی آن بصورت فایل json. صورت می گیرد و برای لود فایل در اپ از require() استفاده می کنیم .

در زیر یک پیکربندی فایل JSON-based قرار دارد

```
{
    "development": {
     "db_host": "localhost",
     "db_user": "root",
     "db_pass": "root"
    },
    "production": {
     "db_host": "10.10.10.10",
     "db_user": "myappdb",
     "db_pass": "!p4ssw0rd#"
    }
}
```

کد فوق را در فایل config.json ذخیره کنید و خط زیر روش لود آن می باشد

```
var config = require('./config.json')[app.get('env')];
```

اپ خفن شما آمده اجرا است اما اگر می خواهید نتیجه خروجی را در ترمینال ببنید کد زیر را اضافه کنید .

```
console.log(config.db_host); // 192.168.1.9
console.log(config.db_user); // myappdb
console.log(config.db_pass); // !p4ssw0rd#
```

اپ شماره هفت

Set و Get/پلکیشن

اپ Express یک تابع پیش تعریف شده به نام set دارد که پیشتر از این در اپ ها استفاده کرده ایم این تابع مقدارهای گوناگون داینامکی را برای app از این طریق می توان مقدار دهی کرد . پیشتر از این این دو نمونه زیر استفاده کرده ایم

```
app.set('view engine', 'jade');
app.set('views', './views');
```

مقدارهای از متغییرهای اپلکیشن را می توان بدست و استفاده کرد از طریق متد app.get() .

جدول زیر لیست از گزینه های که برای پیکر بندی اپ express می توانید از آنها استفاده کنید موجود می باشد

گزینه	معنی و منظور
env	وضعیت در حال اجرا . پیشنهاد می شود بصورت دستی مقدار دهی نشود (در بخش بعد بیشتر خواهید دانست)
trust proxy	فعال کردن پروکسی
json repacer	باز خواندن JSON
jsonp callback name	باز خوانی نام برای درخواست های JSONP
case sensitive routing	حساس بودن بزرگ و کوچک بودن نام route
strict routing	اسلش پایانی در آخر نام یک route
view cache	نمایش پشته
view engine	موتوری برای پروسس فایل ها
views	مسیری از فایل ها

Environment متفاوت اکسپرس

در پروسه تولید نرم افزار سیستم ها توسعه داده می شود به شکلی که برای مراحل تست کاربر پذیری , مرحله تکمیلی و ارائه , تولید و برای ارائه جهت تولید نسخه نهایی . بصورت فنی به این زمینه از اجرای و نرم افزار را وضعیت [19] می گویند .

توضیح بالا روش بسیار عمومی وضعیت نرم افزار است که در اکسپرس روش متفاوتی از روش معمول استفاده شده . برای مثال در وضعیت توسعه سازندگان اکسپرس تمایل دارند تا ببنید درباره جزییات هر کدام از خطاهای نرم افزار را . بریم برای دیدن چگونگی کار کرد آن.

متد اکسپرس app.get('env') وضعیت جاری اپ را بر می گرداند . وضعیت یک اپ را نیز می توان تغییر داد اما پیشنهاد می شود این عمل بصورت دستی انجام نشود .

پیش از آن که تنظیمات پایه ای اپ را بصورت وضعیت انجام دهیم چگونگی کار کرد app.get('env') را توضیح می دهیم

زمانی که یک اپ اکسپرس شروع می شود به دنبال یک متغییر وضعیت می گردد و صدا می زند NODE_ENV از آبجکت procress.env , اگر موفق به یافتن آن شد مقدار NODE_NEV علامتگذاری می شود به متغییر env اپ در صورت یافت نشدن مقدار development علامتگذاری می شود . متغییر اپ توسط app.get خوانده و توسط app.set نوشته می شود .

بصورت خلاصه : اگر NODE_ENV ماشین را تنظیم نکرده باشید وضعیت مقدار development خواهد گرفت . اگر قصد استفاده از یک ماشین توسعه را دارید نیاز به تنظیم کردن و هیچ مقدار دهی نمی باشد در غیر اینصورت ِ بویژه برای تولید محصول ِ درخواستی برای تنظیم اسم توسعه ارسال شود .

پایان

[19] Environment

منابع

1- E-book : Concepts of Programming Languages 10th
2- E-book : Node beginner
3- Link[document]: https://github.com/joyent/node
4- Link [installation and API] http://nodejs.org
5- Link [about node.js] http://css.dzone.com/articles/quick-introduction-how-nodejs
6- https://github.com/joyent/node/wiki/