**Node.js چیست :**

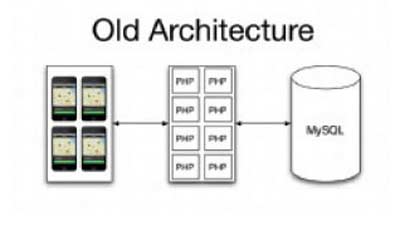
نود، یک سیستم ‌نرم‌افزاری است که برای نوشتن برنامه‌هاي اینترنتی (به‌خصوص وب سرورهای جدید و پیشرفته) با توجه ویژه به قابلیت مقیاس پذیری بالا توسعه‌داده شده است.   
  
در این پلتفرم، برنامه‌ها با زبان جاوا اسکریپت نوشته مي‌شوند که در آن، با استفاده از سیستم ورودی‌/‌خروجی ناهمزمان (Asynchronous) و رویداد محور، سعی در کاهش سربار و افزایش مقیاس پذیری پلتفرم شده است.   
  
این پلتفرم در ابتدا توسط رايان داهل (Ryan Dahl) در سال ۲۰۰۹ نوشته شد و رشد و توسعه آن مرهون پشتیبانی Joynet، شرکت متبوع رایان داهل است.   
  
این پلتفرم با استفاده از زبان C و همچنین خود جاوا اسکریپت نوشته‌شده و حاوی موتور جاوا اسکریپت V۸ گوگل کروم و سیستم ماجولی Common JS به همراه برخی دیگر از کتابخانه‌هاي دیگر است و بر‌خلاف آنچه تا‌کنون مرسوم بود، نه روي مرورگر که در سمت سرور اجرا می‌شود.   
  
با استفاده از این پلتفرم، به سادگی مي‌توان وب‌سرورهای همزمان (Concurrent) بسیاری ایجاد کرد تا عملیات ورودی و خروجی را به روشی پر بازده (که با نام روش non-blocking شناخته مي‌شود) مدیریت ‌کرد.   
  
چنین امکاناتی برای TCP، DNS و HTTP در این پلتفرم در‌نظر گرفته‌شده است.   
  
این پلتفرم به طور کامل حاوی API جاوا اسکریپت بوده و همچنين، بخش‌هاي اضافه‌اي را نیز برای انجام عملیات‌هایی مانند کار با سیستم‌فایلي ارائه مي‌کند.   
  
توجه داشته باشید که Node.js باید به‌صورت جدا دانلود شده و پس از کامپایل در سرور مورد نظر، برای اجرای کدهای جاوااسکریپت مورد استفاده قرار گیرد .

چه کسانی از Node.Js استفاده می کنند ؟

1. YAHOO
2. eBay
3. Linkedin
4. Microsoft

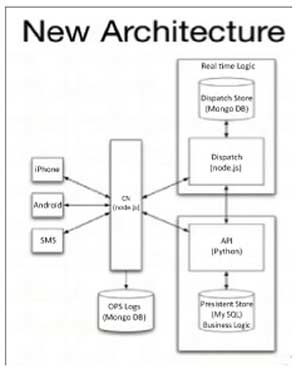
**چرا پلتفرم نود مهم است؟**

با این‌که تاکنون پلتفرم‌هاي مشابهی با زبان‌هاي مختلف در این زمینه معرفی شده‌اند، اهمیت Node.js در این است که رایان داهل و همکاران وی با استفاده از جاوا اسکريپت چنین پلتفرم سمت سروری را تولید کرده‌اند که بی‌اغراق، مورد استفاده تمام توسعه‌دهندگان وب تاکنون بوده است زيرا همه آن‌ها برای انجام امور سمت کلاینت باید از جاوا اسکریپت استفاده مي‌کردند و معرفی نود باعث شد تا بسیاری به این فکر بیفتند که دیگر به زبان سمت سرور اختصاصی خود نیاز نداشته و با استفاده از جاوا اسکریپت، بسیاری از دردسرها کاهش خواهد یافت.   
  
از طرف دیگر، پارادایم خاص جاوا اسکریپت برای کار با Callback‌ها و انجام امور ورودی/خروجی به‌صورت non-blocking نیز در توجه توسعه‌دهند‌گان و گرایش بسیاری به سمت آن تأثیر به‌سزایی داشته است.   
  
برای روشن‌شدن میزان سودمندی نود، مثالي ذکر مي‌کنيم.   
تصور کنید در یک سایت اینترنتی، هزاران کاربر با وارد‌کردن URL سایت وارد آن شده و بسیاری دیگر، درخواست‌هاي AJAX بی شماری به وب سرور سایت مذکور ارسال مي‌کنند.   
  
در این شرایط، هر درخواست یک رشته یا thread جدید در برنامه وب سرور مورد استفاده (که عموماً آپاچی خواهد بود) ایجاد کرده تا اسکریپتی را، براي نمونه PHP که به احتمال، حاوی پرس‌و‌جو از پایگاه‌داده یا چیزی شبیه به آن است، اجرا کند و نتیجه این اجرا را برای مرورگر درخواست‌کننده ارسال کند.   
  
در این کاربرد دو مفهوم اصلی از اهمیت ویژه‌اي برخوردار هستند: یکی حافظه و دیگری ورودی/خروجی بلوکه‌کننده. زيرا هر thread اجرایی، حافظه بسیار زیادی را اشغال‌کرده و تعدد کاربران، امکان پر‌شدن حافظه سیستم و از کار افتادن خدمات مربوط را پیش‌مي‌آورد.   
  
همچنين، هر کدام از این رشته‌هاي پردازشی به عملیات ورودی/خروجی مانند دسترسی به پایگاه‌داده نیاز دارند که هرکدام به زمانی مشخص برای تکمیل‌شدن نیازمند است.   
  
در بيشتر موارد نیز به دلیل بزرگ‌بودن جدول‌هاي داده‌اي و نتایج پرس‌و‌جوهای انجام شده، اسکریپت اجرا‌کننده مجبور است تا باز‌گردانی داده‌ها از طرف پایگاه‌داده منتظر‌مانده و پس از دریافت، به اجرای بقیه دستورها بپردازد.   
  
به همین دلیل، این عملیات، عملیات ورودی/خروجی بلوکه کننده نامیده مي‌شود و تأثیری بسیار منفی روی عملکرد کلی سرور خواهد داشت. نکته منفی این رویداد علاوه بر انتظار اسکریپت برای بازگردانی داده‌هاي درخواست شده، مصرف توان‌پردازشی و حافظه در‌حین انتظار است که در صورتی که در مقیاس بالا اتفاق بیفتد، باعث نا‌کار‌آمدی کلی سیستم خواهد شد (شکل‌۲).



شکل ۲- معماری گذشته وب‌سرورها

حال اگر راهی باشد که بتوان این رشته‌هاي پردازشی را حذف کرده و همه را در یک انباره بزرگ قرار داد، مشکل مصرف حافظه برطرف خواهد شد.   
  
همچنين، در صورتی که بتوان تا زمانی که برای بازگردانی نتایج ورودی/خروجی (پرس‌و‌جو از پایگاه‌داده) منتظر هستیم، به انجام امور دیگر در اسکریپت خود بپردازیم مشکل بلوکه شدن نیز حذف شده و سرعت عملکرد به طور قابل توجهی افزایش خواهد یافت.   
  
این دو مورد دقیقاً کارهایی است که Node.js به انجام آن‌ها مبادرت ورزیده و به همین دلیل، اهميت بالایی در آینده برنامه‌هاي کاربردی وب خواهد داشت.   
  
یک سرور مبتنی بر Node.js، تمام اتصال‌هاي ورودی را در یک انباره مدیریت کرده و فریم‌ورک پس زمینه نود، مي‌تواند حالات و شرایط این اتصال‌ها را مدیریت کرده و براي نمونه، یک اتصال را به حالت «در حال انتظار» تبدیل کند.   
همچنين، در پلتفرم نود از پارادیم حلقه رویداد و Callback برای انجام امور پرس‌و‌جو از پایگاه داده استفاده شده است تا انتظار برای پاسخ از پایگاه داده، منابع ارزشمند سیستم را بلوکه نکند.   
  
در این حالت، فرآیند Callback به داده‌هاي بازگشتی از پایگاه‌داده تخصیص داده شده وهمزمان، نود به انجام امور دیگر مي‌پردازد که خود از هدر رفتن سیکل‌های CPU بیشتری جلوگیری مي‌کند.   
با استفاده از این روش، مرورگر سمت کلاینت تا فراهم‌شدن نتایج در حالت انتظار باقی‌مي‌ماند و با حالت درخواست داده‌ها از سرور threaded سنتی‌، هیچ تفاوتی احساس نمی‌کند.   
  
پس از آماده شدن نتایج پرس‌و‌جو از پایگاه‌داده، کد مشخص شده به‌عنوان Callback اجرا شده‌، اتصال مربوط را از انباره انتخاب‌کرده و حالت آن را از وضعیت «درحال انتظار» خارج کرده و با استفاده از آن، داده‌هاي لازم را به مرورگر ارسال می‌کند. استفاده از چنین ساختاری علاوه بر کاهش مصرف حافظه، سرعت اجرای بالایی را نیز به ارمغان مي‌آورد (شکل۳).



شکل ۳- معماری قدیمی، اکنون با استفاده از node.js تغییر یافته است.   
  
همان‌طور که مي‌دانید، برنامه‌هاي تحت وب امروزی که بر مبنای AJAX کار مي‌کنند، بار کاری سنگینی را به سرورهای خود تحمیل مي‌کنند.   
  
در صورتی که به برنامه‌هاي گفت‌وگو یا بازی‌هاي تحت مرورگر چند بازیکنی توجه کنیم، متوجه خواهیم شد که آن‌ها اتصال‌هاي همزمان بسیاری را ایجاد مي‌کنند که زمان کم پاسخ‌گویی به هر کدام از آن‌ها بسیار مهم و حیاتی است.   
  
در چنین کاربردهایی است که قوت Node.js مشخص شده و روش مورد استفاده آن برای مدیریت امور بسیار پر اهمیت‌تر از گذشته به‌نظر مي‌آید.   
  
اگرچه تلاش‌هاي انجام شده برای استفاده از AJAX و Comet برای بهتر‌کردن تجربه وب بسیار ارزنده بوده و سوکت‌هاي HTML5 نیز تلاش قابل تقدیری برای توسعه و ایجاد سهولت در پاسخ به این نیازمندی‌ها بودند، اما تکمیل‌کننده این حلقه، استفاده از فناوري‌هاي جدید در سمت سرور بود که با معرفی و توسعه روزافزون Node.js به خوبی در حال شکل‌گیری است (شکل‌۴).



شکل ۴- جایگاه کنونی نود در بازار سرورها   
  
در هر صورت، برای مقبولیت و گسترش روز‌افزون Node.js به اندکی زمان نیاز است تا توسعه‌دهندگانی که در چند سال اخیر و تا‌کنون بر مبنای پارادایم‌هاي سنتی AJAX به توسعه کد مي‌پرداختند، با روش جدید تطبیق پیدا کرده و به سوی آن گرایش پیدا کنند.   
  
زمانی که سایت‌هاي معروف و برنامه‌هاي وب مبتنی بر AJAX هر کدام سرور اختصاصی خود را داشته باشند، چندان دور نیست.   
  
در صورتی که تجربه کار با AJAX، COMET یا جاوااسکریپت را داشته‌اید، تجربه کار با Node.js را از دست ندهید.   
  
مطمئن باشید که پس از آن، هیچ علاقه‌اي برای بازگشت به استفاده از PHP نخواهید داشت.

مثالی از Nod :وب سرور

این وب سرور ساده به تمام درخواست ها با پیام "hello world" پاسخ می دهد.

var http = require('http');

http.createServer(function(req,res){

res.writeHead(200,{'Content-Type':'text/plain'});

res.end("Hello World");

}).listen(1337, "127.0.0.1");

console.log("Server running at <http://127.0.0.1:1337/>");

برای اجرای این وب سرور کافی است کد بالا را در فایلی با نام دلخواه مثلا example.js قرار دهید و توسط node آن را اجرا کنید:

$ node example.js

Server running at <http://127.0.0.1:1337/>

به عنوان مثالی دیگر می توان سرور TCP ایجاد کرد که تمام درخواست ها را echo می نماید:

var net = require('net');

var server = net.createServer(function(socket){

socket.write("Echo Server:\r\n");

socket.pipe(socket);

});

server.listen(1337,"127.0.0.1");

هدف نود :

هدف ند، بدست دادن راهی ساده برای ساخت برنامه های قابل گسترشِ تحت شبکه است. در مثال وب سرور "Hello World" که در بالا از نظر گذشت، تعداد زیادی از کانکشن ها به طور همزمان قابل پردازش اند. ند ( از طریق epoll، kqueue، /dev/poll، یا select ) به سیستم عامل فرمان می دهد که هر زمان کانکشن جدیدی ایجاد شد، وی را باخبر سازد. سپس به حالت sleep می رود. اگر اتصال جدیدی برقرار شد، سیستم عامل callback را صدا می زند. هر کانکشن در واقع تنها یک اختصاص دهی کوچکِ Heap است.

این روش با مدلهای همزمانی (concurrency) کنونی که در آن از Thread های سیستم عامل استفاده می شود متفاوت است. برنامه های تحت شبکه که بر پایه Thread ها ایجاد می شوند ناکارآمد و برای استفاده، پیچیده اند.

ند در استفاده بهینه از حافظه تحت فشارهای بالا نسبت به سیستم هایی که برای هر کانکشن ۲ مگابایت thread stack اختصاص می دهند، بسیار بهتر عمل می کند. گذشته از این، کاربرانِ ند دیگر نگرانی در مورد dead-lock ها ندارند—هیچ قفلی وجود ندارد. بعلاوه هیچ دستوری در ند مستقیماً کار I/O را انجام نمی دهد، بنابراین پردازه هیچگاه بلاک نمی شود. و چون هیچ چیزی بلاک نمی شود برنامه نویسان غیر حرفه ای هم می توانند سیستم های پرسرعتی بنویسند.

ند در طراحی تحت تأثیر و بسیار شبیه سیستم هایی نظیر Event Machine رابی و یا Twisted پایتن (Python) است. ند مدل رویدادگرا را کمی جلوتر می برد— در ند چرخه ی رویداد به عنوان یک کتابخانه جدا از زبان نیست، بلکه با زبان گره خرده است. در سیستم های دیگر یک blocking call برای شروع چرخه رویداد وجود دارد. متد رایج این است که callback های مورد نیاز در شروع اسکریپت نوشته می شوند و در پایان سرور از طریق اجرای یک متدِ بلاک کننده یا Blocking Call شروع به کار می کند. مثلاُ در رابی داریم: EventMachine::run(). در ند چنین متدی برای شروع حلقه رویداد وجود ندارد. ند به سادگی پس از پردازش اسکریپت ورودی به چرخه ی رویداد وارد می شود. ند با تشخیص اینکه دیگر هیچ callback ای برای اجرا وجود ندارد از این حلقه خارج می شود. این عملکرد مانند اجرای جاوا اسکریپت در مرورگرهاست. حلقه رویداد از دید کاربر مخفی است.

در ند HTTP یک پروتکل ممتاز است. این کتابخانه بر اساس تجربیات نویسنده آن در توسعه برنامه های تحت شبکه و وب سرورها ایجاد شدهاست. برای مثال data streaming در اکثر framework ها کاری غیر ممکن است. ند سعی کرده است که این مشکلات را در پارسر HTTP خود و API های مربوط به آن حل کند. این ویژگی به همراه ساختار رویدادگرای ند، می تواند مبنایی برای بهترین کتابخانه ها و Framework های تحت وب باشد.

اما همزمانی در سیستم های چند پردازنده ای چطور؟ آیا thread ها برای گسترش برنامه روی کامپیوتر های چند هسته ای ضروری نیستند؟ در ند می توانید از دستور child\_process.fork() برای شروع پروسس های جدید استفاده کنید. این پروسس های جدید به صورت موازی اجرا خواهند شد. همچنین می توانید از ماژول cluster به منظور load balancing کانکشن ها روی چند پروسس استفاده کنید.

همچنین ببینید:

**Node.js در یک نگاه**   
توسعه وب با استفاده از یک زبان داینامیک (جاوااسکریپت) که روي یک ماشین مجازی خیلی سریع با نام V۸ اجرا مي‌شود.   
  
سرعت اجرای آن از Ruby ،Python و Perl بسیار بیشتر است. توانایی مدیریت‌کردن هزاران اتصال همزمان، با کمترین سربار و با استفاده از یک نرم‌افزار.   
  
شایستگی ذاتی جاوااسکریپت در کار با event loop‌ها با داشتن اشیاي درجه یک، قابل استفاده در توابع و همچنین نحوه بستن‌آن‌ها که سال‌ها توسط برنامه‌نویسان مختلف روي مرورگرها تمرین شده است.   
  
تعدد افرادی که برنامه‌نویسی با استفاده از جاوااسکریپت را مي‌دانند یا در آن حرفه‌اي هستند. به جرأت مي‌توان جاوااسکریپت را محبوب‌ترین زبان برنامه‌نویسی حال حاضر دانست.   
  
استفاده از جاوااسکریپت در سمت سرور به همراه استفاده از آن در سمت کلاینت، احتمال ناهمخوانی و بروز مشکلات و معضلات محیط‌های ناهمگون برنامه‌نویسی را کاهش‌داده و امکان برقراری ارتباط داده‌اي با استفاده از JSON میان هر دو طرف را فراهم مي‌سازد. استفاده از یک کد اعتبار‌سنجی فرم چه در سمت سرور و چه در سمت کلاینت واقعاً لذت‌بخش است.

توجه : برای نرم افزار های ساده با تعداد کاربر کم پیشنهاد نمیشود ، ولی برای برنامه هایی که بسیار پر هزینه از نظر اتصالات کاربران هستن (مثل پیام رسان یاهو) این تکنولوژی فوق العادس  
  
**مزایا**   
- مقیاس‌پذیر به هزاران اتصال فعال   
- بسیار سریع (به خصوص در مقایسه با PHP و Ruby)   
- امنیت بیشتر در مقابل بار اضافی اعمال شده به سرور (به خصوص در زمان وقوع حمله‌هاي DDOS که بقیه نرم‌افزارهاي سرور مانند SSH قابل دسترسی و پاسخ دهنده باقی مي‌مانند)   
- پارادایم ناهمزمان بسیار ساده، جذاب و آشنا   
- نیاز نداشتن به درگیری با مسائلی مانند thread-safety   
- عدم استفاده از Multithreading و به تبع آن، مواجه نشدن با باگ‌های قفل‌کننده   
- جامعه توسعه‌دهندگان بسیار عظیم و کتابخانه‌ها و ابزارهای توسعه‌داده شده غنی به همراه یک Package Manager قوی برای مدیریت ملحقات!   
  
**معایب**   
- پیچیدگی کد برنامه ناهمزمان (Asyncronous) و مشکل بودن یادگیری مفاهیم اولیه برای برنامه‌نویسان نا‌آشنا با مدل برنامه‌نویسی موازی.   
- تأخیر بسیار بالا در صورت نیاز وظایف داخلی به اتمام عملیات محوله. به دلیل این‌که برنامه به‌صورت تک رشته‌اي اجرا مي‌شود، یک تابع با زمان اجرای طولانی مي‌تواند سرعت پاسخ‌دهی کلی سیستم را به شدت کاهش دهد.   
- نبود یک کتابخانه استاندارد جاوااسکریپت. برنامه‌نویسان جاوااسکریپت به شدت به استفاده آسان از آن عادت کرده‌اند، بدون این‌که به وارد کردن کتابخانه خاصی نیاز داشته باشند. به‌همین دلیل، از هر چیزی در برنامه‌ها پنج مدل مختلف وجود خواهد داشت که سردرگمی خاصی را ایجاد مي‌کند. حتی ماجول‌هاي قرار‌داده شده در هسته Node.js نیز هر کدام پنج نوع مختلف دارند که به تکامل سریع‌تر مي‌انجامد، اما درجه ابهام بالایی را نیز در پی‌خواهد داشت.   
- سیستم‌هاي نهایی نوشته‌شده با این فناوري به‌شدت نسبت به مدل CGI، یعنی Apache+PHP یا Perl یا Ruby و... پیچیده‌تر بوده و استثناهای مدیریت‌نشده مي‌توانند کل فرآيند را متوقف‌کرده و نیاز به راه‌اندازی دوباره فرآيندهاي درحال کار روي کلاستر را الزامی سازند. یک کد باگ دار نیز مي‌تواند باعث خرابی فرآيند اجرایی مربوط شده و هر فرآيند درحال کار خراب، درخواست‌هاي بسیاری را بی‌پاسخ خواهد‌گذاشت که به تبع آن مقاومت کل سیستم در مقابل خرابی را کاهش داده و باعث کاهش کیفیت خدمات خواهد شد.

یه مثال از کاربرد Node.Js :

مثلاً فکر کنید می خواهیم یک برنامه چت رو بنویسیم، وقتی ما یک پیغام را تایپ می کنیم، با فشردن دکمه Enter متن ما برای سرور ارسال می شه، اما وقتی طرف مقابل ما برای ما یک پیغام را تایپ می کنه سیستم ما متوجه نمی شه که از طرف مقابل پیغامی ارسال شده، یکی از راه هاش اینه که ما یه تایمر بذاریم و هر چند ثانیه چک کنیم که پیغام جدید ارسال شده یا نه ،به نظر شما گذاشتن تایمر راه خوبیه ؟

حالا فکرش رو بکنید که ۱۰۰ نفر همزمان با هم بخوان چت کنن و در هر ثانیه چندیدن درخواست به سرور ارسال می شود که اطلاعات را برای ما بفرسته، در نتیجه بار سنگینی به سرور تحمیل می شه و دیگه لازم نیست کسی سرور شما رو هک کنه، شما خودتون زحمت Down کردن سرور رو کشیدید !!!

NodeJS می تونه کاری انجام بده که این ارتباط به صورت دو طرفه برقرار بشه، یعنی اینکه سرور هر موقع خواست برای شما دیتا ارسال کنه، حالا چه اتفاقی می افته ؟  
  
به محض اینکه شخص مقابل شما یک پیغام برای شما تایپ می کنه، این پیغام برای شما پدیدار می شود !  
  
حالا چرا تایمر سیستم رو منفجر می کنه ولی NodeJS فشاری نمیاره ؟

شما در Node یک پورت را باز نگه می دارید و پورت رو به اصطلاح گوش می دهید و هر دیتایی که روی پورت مورد نظر ارسال بشه شما روی آن پردازش انجام می دهید، پس دیگه نیازی نیست پشت سره هم به سرور درخواست ارسال کنید.

**NodeJS چطوری نصب می شود؟**  
شما می تونید به سایت Nodejs.ir سر بزنید و آخرین ورژن NodeJS رو با توجه به سیستم عامل خودتون دانلود و نصب کنید، با توجه به اینکه من بشخصه ویندوز رو دوست دارم، نصب در سیستم عامل ویندوز رو توضیح می دم و لینوکس دردسر زیاد داره !  
  
خوب بعد از دانلود، برنامه رو نصب کنید.

استفاده از node.js در ویندوز :

نصب و راه اندازی nodejs بر روی سیستم عامل ویندوز بسیار ساده است. در نسخه های بالاتر از ورژن 0.6.3، installer برنامه npm رو هم به طور خودکار برای شما نصب می کنه.

می تونید آخرین نسخه ند رو از آدرس http://nodejs.org/dist/latest دریافت کنید. دو نسخه برای ویندوز آماده شده. یکی فایل اجرایی و یکی فایل msi.

فایل installer یا همون msi رو دانلود و اجرا کنید. این نصاب سه کار انجام میده. اول برنامه ند رو داخل یک پوشه کپی می کنه. بعد npm رو هم نصب می کنه. و در آخر آدرس ند رو در متغیر محیطی PATH قرار می ده تا بتونید از node در command prompt استفاده کنید. به همین دلیل نیاز دارید تا یک بار سیستم رو reboot کنید.

نصب Manual

راه دیگه استفاده از فایل اجرایی node هست. این فایل رو از همون آدرس قبلی دانلود کنید و در پوشه ای ذخیره کنید. مثلا پوشه c:\bin\node. این آدرس رو به این صورت در PATH اضافه کنید:

Control Panel > System and Security > System > Advanced System Settings

در داخل پنجره ای که باز می شه، روی تب Advanced کلیک کنید و بعد روی دکمه Envirnoment Variables کلیک کنید. حالا در قسمت پایین پنجره Scroll کنید تا متغیر PATH رو ببینید. این متغیر رو به این صورت تغییر بدید که در انتها بعد از اضافه کردن ";" آدرس جایی که node در اون قرار داره رو اضافه کنید.

تنها مشکل نصب دستی node اینه که باید خودتون npm رو نصب کنید! البته این کار زیاد سخت نیست. از آدرس http://npmjs.org/dist آخرین ورژن npm رو هم به صورت zip دانلود کنید و در پوشه ای که node رو نصب کردید extract کنید.

حالا باید همه چیز آماده باشه.

تست

Command Prompt رو اجرا کنید. با تایپ node باید محیط interactive ند اجرا بشه. اگر با خطای پیدا نشدن این دستور مواجه شدید یک جایی در تنظیم متغیر PATH مشکل وجود داره.

برای تست npm هم کافیه داخل یک پوشه مثلا test، یک ماژول مثل express رو نصب کنید:

npm install -g express

اگر این مرحله هم با موفقیت انجام شد همه چیز برای یک تجربه لذت بخش با ند آماده است!

جامعه ند

با وجود جوان بودن Node.js، جامعه آن بسیار بزرگ و متشکل از افراد خلاق و دوست داشتنی است. در اینجا سعی کرده ایم بهترین فروم های مرتبط را معرفی و کمی هم در مورد جامعه ند فارسی صحبت کنیم.

گاهنامه ها

How To Node مشتمل بر تعداد زیادی از مقالات کاربردی در رابطه با ند است. Planet Node یک aggregator برای وبلاگ های مرتبط با ند است. Nodeup پادکست های هفتگی در ارتباط با آخرین اخبار جامعه ند منتشر می کند. وبلاگ جامعه ند فارسی هم به نشانی nodejs.ir/blog در خدمت شماست!

گیتهاب

گسترش ند به صورت متن باز در مخزن nodejs در گیتهاب صورت می پذیرد. معمولاً کامنت‌های مرتبط با هر commit پر از بحث های داغ در ارتباط با کدنویسی و گسترش هسته ند است. صفحات ویکی مخزن منابع خوبی برای شروع کار با ند در اختیار شما قرار می دهد. حتماً به صفحات پروژه‌ها، برنامه‌ها و شرکت‌های استفاده کننده از ند و همچنین لیست بلندبالای ماژول‌های نوشته شده مراجعه کنید! بسیاری از این ماژول‌ها در npm نیز منتشر شده اند.

باگ‌ها و مشکلات به آدرس https://github.com/joyent/node/issues گزارش داده می‌شوند. همچنین از مشارکت شما در رفع مشکلات استقبال می‌گردد. برای اطلاع از نحوه مشارکت در گسترش ند به راهنمای مشارکت مراجعه کنید.

Mailing Lists

دو Mailing List اصلی برای ند در نظر گرفته شده است: یکی مربوط به کاربران که در آن به معرفی ماژول‌ها و بحث و تبادل نظر پیرامون ند و ... پرداخته می‌شود. دیگری Mailing List مربوط به توسعه هسته ند است که در مورد طراحی داخلی ند و ویژگی‌های آینده آن بحث می‌شود.

جامعه ند فارسی نیز دارای یک Mailing List به آدرس http://groups.google.com/group/nodejs-fa می‌باشد که در آن به بحث و تبادل نظر در ارتباط با ند می‌پردازیم.

IRC

برای چت کردن در مورد ند به آدرس irc.freenode.net، کانال #nodejs مراجعه کنید. می توانید از کلاینت IRC استفاده کنید و یا از برنامه تحت وب freenode استفاده نمایید. Felix Geisendorfer از اعضای هسته اولیه ند، لاگ روزانه IRC را نگهداری می کند.

کنفرانس ها

NodeConf سلسله کنفرانس‌های اصلی ند در ایالات متحده است که توسط Mikeal Rogers مدیریت می‌شود. گروه کاربران ژاپنی ند هم کنفرانسی تحت نام NodeFest برگزار می‌نمایند. کنفرانس NodeCamp.de نیز آلمان توسط Rails Love برگزار می‌شود. کنفرانس ایتالیایی ند هم در این لیست جای می‌گیرد. کنفرانس Node Summit کنفرانسی است که متمرکز بر استفاده ند در شرکت‌های بزرگ است و در سانفرانسیسکو برگزار می‌شود. البته در کنفرانس JSConf که کنفرانس اصلی Javascript است ارائه‌هایی در خصوص ند وجود دارد.

جامعه ند فارسی

و اما جامعه ایرانی ند! این گروه هنوز در اول راه است. سعی کرده‌ایم همگام با جامعه جهانی ند قدم برداریم و از تکنولوژی روز عقب نمانیم. بنابراین تلاش کردیم سایت فارسی ند را به روز و تر و تازه نگهداریم. چندین بار نام جامعه فارسی و سایت فارسی ند در کنفرانس‌های بین المللی و از زبان سازنده ند برده شد و مورد تشویق قرار گرفت.

این سایت و وبلاگ مربوطه به صورت گروهی مدیریت می‌شود و افراد علاقه‌مند می‌توانند مقالات خود را برای ما ارسال نمایند تا با نام خودشان در وبلاگ درج شود. علاوه براین از هر نوع اصلاح و ویرایش که منجر به بهبود سایت گردد استقبال می‌کنیم. اطلاعات بیشتر راجع به نحوه مشارکت را در این صفحه ببینید. با عضویت در Mailing List گروه، نظرات و پیشنهادات خود را مطرح نمایید و در بحث‌ها مشارکت کنید.

منابع :

[http://www.itna.ir](http://www.itna.ir/)

<http://nodejs.ir/>