**به نام خدا**

**NodeJS چیست؟**

* **یک محیط است برای اجرای کدهای جاوا اسکریپت یعنی runtime environment. (یک فریم ورک یا زبان برنامه نویسی نیست)**
* As an **asynchronous** **event-driven** JavaScript runtime, Node.js is designed to build scalable network applications.
* قبلا هر مرورگری برای خود محیط اجرای جاوا اسکریپت داشت و وظیفه ی اجرای کدها با خودش بود. در سال 2009
* شخصی با یک ایده، این محیط اجرایی را از دل مرورگر خارج کرد و برای این کار از موتور V8 که همان محیط اجرایی جاوا اسکریپت گوگل کروم است، استفاده کرد.



* Node.js به صورت single thread کار می کند. بنابراین برای برنامه های cpu intensive مناسب نمی باشد.
* محیط اجرایی نود، REPL است. Read, eval, print, loop
  + این محیط دقیقا مشابه ابزار console در مرورگر است.
  + وقتی در این محیط دو بار Tab زده شود تمام دستورات لیست می شوند. اونایی که با حروف بزرگ شروع شده اند مربوط به خود جاوااسکریپت هستند ولی حروف کوچک مختص Node.js است.
* متغیرهای سراسری: متغیر سراسری در مرورگر window و در nodejs عبارت global می باشد.
* به هر فایل در node یک ماژول گفته می شود.
  + هر متغیری که تعریف می شود به صورت پیش فرض، فقط در همان ماژول قابل دسترس خواهد بود.

ماژول ها:

* Core Module: ماژول هایی هستند که خود Node.JS توسعه داده. مثل ماژول os.
* Local Module: ماژول هایی هستند که داخل برنامه توسط برنامه نویس تعریف می شوند.(فایل های برنامه). معمولا ماژول اصلی برنامه اسمش app.js هست.
* Third Party Module: ماژول هایی هستند که توسط npm به برنامه اضافه می شوند. (ماژول های نوشته شده توسط دیگران)
* COMMONJS: نام پروژه ای است که برای کار با ماژول ها استفاده می شود.
  + برای استفاده از متغیرها و توابع در خارج از ماژول، باید آنها را export کرد و در ماژول مقصد هم باید آنها را require کرد.

Module.exports.[custom name] = [variable or function name]

* Module Wrapper Function: تمام کدهایی که در ماژول وجود دارند داخل این تابع احاطه می شوند:

(function (exports, require, module, \_\_filename, \_\_dirname)){

…

})

* برای استفاده از ES6 در NodeJS باید ابتدا npm init اجرا شود تا فایل package.json ایجاد شود. سپس خط زیر را اضافه می کنیم:

“type” : “module”,

* + سپس برنامه را با دستور node –experimental-modules [app.js] اجرا می کنیم.
  + در این حالت از عبارات import و export برای ارتباط بین ماژول ها استفاده می کنیم.
  + گفتنی است که import به صورت asynchronous و require به صورت synchronous کار می کند.
* Node-modules: فولدری است که فایل های داخل آن از طریق npm نصب می شوند. و اطلاعات تکمیلی آن داخل فایل package.json قابل مشاهده است.

مثال: ایجاد فولدر در مسیر جاری:

const fs = require('fs');

const path = require('path');

fs.mkdir(path.join(\_\_dirname,'test'), {}, function(err){

    console.log(err);

})

یا

fs.mkdir(path.join(\_\_dirname, 'test'), {}, (err)=>{

    if (err) console.log(err);

    console.log('The folder is created.');

})

EventEmitter: ایجاد event و فراخوانی آن

const EventEmitter = require('events');

const emitter = new EventEmitter();

//Event listener

emitter.addListener('messageLog', ()=> {

    console.log('event occured.');

})

Or

emitter.on('messageLog', ()=> {

    console.log('event occured.');

})

//Raise the event

emitter.emit('messageLog');

Event call with arguments type 1:

//Event listener

emitter.addListener('messageLog', (name, family)=> {

    console.log('event occured.'+ name + " " + family);

})

//Raise the event

emitter.emit('messageLog', 'Hesam','Shahbazi');

Event call with arguments type 2:

//Event listener

emitter.addListener('messageLog', (...args)=> {

    console.log('event occured.'+ args[0] + " " + args[1]);

})

//Raise the event

emitter.emit('messageLog', 'Hesam','Shahbazi');

Another sample:

(logger.js)

const EventEmitter = require("events");

class Logger extends EventEmitter{

    log(message){

        console.log(message);

        this.emit('LogMessage');

    }

}

module.exports.Logger = Logger;

(application.js)

const logger = require('./logger');

const Logger = new logger.Logger;

Logger.on('LogMessage', ()=>{

    console.log('Event Occured.');

})

Logger.log("Hesam calling...");

Webpack: وقتی فایل هایی که قرار است مرورگر بارگذاری نماید تا محتوای وب را نمایش دهد کم باشد مشکلی ایجاد نمی شود ولی اگر تعداد این فایل ها زیاد باشد، مرورگر قادر به هندل کردن شرایط نیست. اینجاست که WebPack وارد میدان شده و با تجمیع فایل ها و وابستگی های آنها، تعداد فایل های کم و تجمیع شده را به مرورگر ارسال می کند.