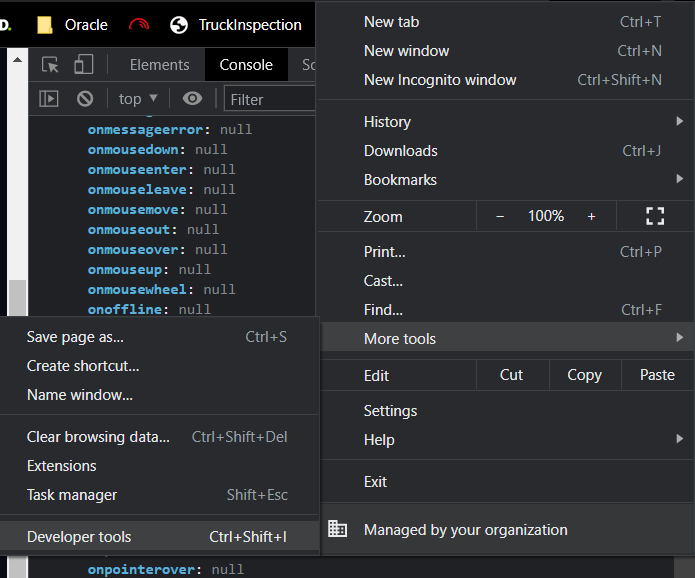
NODE JS TUTORIAL

* To install nodejs -> go to nodejs official website and install latest version.
* To check version of nodejs : enter in CMD -> type **node -v** press enter.

To use Javascript in browser -> click on … of chrome browser and click on More tools -> Deveeloper tools -> then we can use javascript codes in console window.



**Blocking vs Non-Blocking I/O** : blocking I/O(input output operations) means it blocks the entire process while doing any input output operations whereas non-blocking I/O doesn’t blocks the process while performing any i/o operations.

**Documentation of API** : <https://nodejs.org/dist/latest-v16.x/docs/api/>

**Defining function and importing in another module** : utils.js

console.log('util.js')

my\_name\_1 = 'my name from utils'

my\_name\_2 = 'my another name from utils'

// define our fucntion

function add(a,b){

    return a+b

}

function minus(a,b){

    return a-b

}

// export the function add and minus

module.exports =  {add, minus}

App.js

const a = require('./utils')  // import the function utils

const sum = a.add(2,10)

const sub = a.minus(10,4)

console.log(sum)

console.log(sub)

npm 🡪 it stands for NODE PACKAGE MANAGER. It have number of predefined packages that we can use to our projects. The link is <https://www.npmjs.com/>

**Steps to install a node module** : npm install [packageName@10.3.0](mailto:packageName@10.3.0) or npm I [packageName@32.2.1](mailto:packageName@32.2.1)

**Whenever we install a depenedcny we get node\_modules(which has all the packages) and the node module also get added to our package**.json file show below

  "dependencies": {

    "validator": "^13.7.0"

  }

To import node module in our app use :

var validator = require('validator');

**to validate a string is email or not**

console.log(validator.isEmail('andre@gmail.com')) // validates whether it's email or not : TRUE

console.log(validator.isEmail('satish')) // FALSE

**npm install** 🡪 to install all the packages by on its own. The node looks in package.json and installs all the required modules on its own.

**COMMAND LINE ARGUMENTS**

We can access the command line arguments using **process.argv** : it return array of items. Where the first is location of node executables, the 2nd is location of our app.js and then we get our inputs from cmd terminal.

To get individual item we us : **process.argv[n]**

// getting inputs from command line args

console.log(process.argv) // print all command line arguments

console.log(process.argv[2]) // prints the 2nd item

console.log(process.argv[3]) // prints the 3rd item

run this : node app.js satish kumar

o/p:

[

'C:\\Program Files\\nodejs\\node.exe',

'C:\\Users\\satissingh\\Documents\\Important\\NodeJS Tutorial\\Codes\\notes-app\\app.js',

'satish',

'kumar'

]

satish

kumar

**JAVASCRIPTS OBJECT NOTATION :**

const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue",

fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};

**ACCESSING OBJECT PROPERTY:**

*objectName.propertyName* OR *objectName["propertyName"]*

**ACCESSING FUNCTION :** *objectName.methodName()*

**FILTER FUNCTION IN JS** :

filter() creates a new array filled with elements that pass a test provided by a function.

filter() does not execute the function for empty elements.

filter() does not change the original array.

const ages = [32, 33, 16, 40];  
const result = ages.filter(checkAdult);  
function checkAdult(age) {  
  return age >= 18;  
}

**LIST OF MODULES**

**chalk** 🡪 used to make terminal text colourful

**validator** 🡪 to perform different type of validation like url,email and others

**nodemon** 🡪 nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected. To install it use npn i nodemon -g hit enter

**-g stands for global.**

To verify nodemon installation : nodemon -v

**To run a file using nodemon** : nodemon filename.js : the nodemon keeps running and will execute file when we make changes and save it to our file.

**YARGS** : by parsing arguments and generating an elegant user interface.

It helps us in grabbing command line arguments easily without much parsing. We can pass key value pair easily and grab in our application.

yargs = require('yargs')

// get all command line arguments

console.log(yargs.argv) // node fileName.js satish kumar

o/p: { \_: [ 'satish', 'kumar' ], '$0': 'yargs\_demo.js'

console.log(yargs.argv) // node fileName.js satish --title="kumar" ::

op :{ \_: [ 'satish' ], title: 'kumar', '$0': 'yargs\_demo.js' }

// to access our custom key value

console.log(yargs.argv.title)

we can define our own command in yargs.command function and when we use that in cmd terminal that particular function is called.

// we can use yargs command to define our custom command and it will call the respective version of that.

yargs.command({

    command: 'add',

    describe: 'Add a new note',

    handler : function(){

        console.log('Called Add Function')

    }

})

// using yargs module

console.log(yargs.argv)

--help

app.js [command]

Commands:

app.js add Add a new note

Options:

--help Show help [boolean]

--version Show version number [boolean]

To call our function we can use : node app.js add ENTER : it will call the function mapped to add

asdga