Node.js Introduction

Node.js

- Το Node.js αποτελεί ένα **περιβάλλον εκτέλεσης JavaScript** βασισμένο στην ανοιχτού κώδικα v8 Javascript engine της Google.
  (asynchronous event-driven JavaScript runtime environment)

- Μέχρι το 2009, ο μόνος τρόπος εκτέλεσης της javascript ήταν μέσα σε ένα πρόγραμμα περιήγησης (browser).

- Το Node.js περιέχει μια μηχανή javascript, η οποία επιτρέπει την εκτέλεση κώδικα javascript
  - Το Node.js **δεν** είναι γλώσσα προγραμματισμού

Node.Js:
Javascript Outside Of The Browser

Browser

Node.js

Node.js is like a container where javascript can be executed. Outside of any browser.

Node.js pros

1. Node.js σου επιτρέπει να χρησιμοποιείς JavaScript τόσο στον server όσο και στον client. Αυτό σημαίνει ότι μπορείς να χρησιμοποιήσεις την ίδια γλώσσα και για τα δύο, απλοποιώντας την ανάπτυξη και τη συντήρηση του κώδικα.

2. **Υψηλή Απόδοση**: Χάρη στη χρήση του V8 JavaScript Engine της Google, το Node.js είναι πολύ γρήγορο στην εκτέλεση κώδικα JavaScript. Αυτό συνεισφέρει σε μια ταχύτερη επεξεργασία δεδομένων και απόκριση στις αιτήσεις των χρηστών.

3. **Μεγάλη Κοινότητα** Το Node.js έχει μια πολύ μεγάλη κοινότητα προγραμματιστών και ένα εκτενές οικοσύστημα βιβλιοθηκών και εργαλείων. Το npm (Node package manager) είναι η μεγαλύτερη βιβλιοθήκη πακέτων λογισμικού στον κόσμο, προσφέροντας πλήθος εργαλείων και βιβλιοθηκών που καθιστούν την ανάπτυξη πιο γρήγορη και αποτελεσματική.

4. **Ευελιξία στην Ανάπτυξη**: Node.js επιτρέπει την ανάπτυξη διάφορων τύπων εφαρμογών, από web applications και REST APIs μέχρι και real-time systems όπως online games και chat applications.

Node.js pros

- Το Node.js είναι ιδανικό για ανάπτυξη γρήγορων και επεκτάσιμων εφαρμογών που περιλαμβάνουν μεγάλο όγκο δεδομένων.
- Βοηθά στη καλύτερη απόδοση και συνολική παραγωγικότητα προγραμματιστή
- W3schools: Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

Examples of using node.js

- Για APIs με βάση δεδομένων
  - add, delete, modify data
- Chat web apps
- Server –side web apps

Don’t use Node.js when

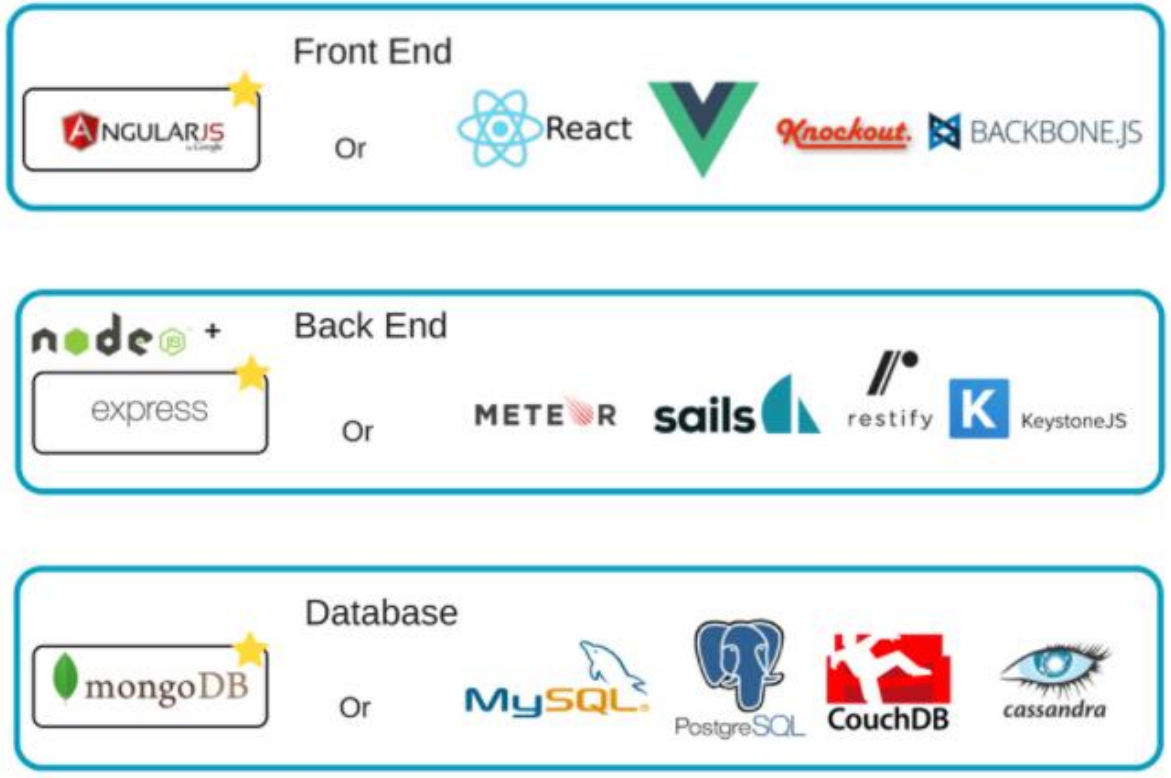
- Don’t use: εφαρμογές με heavy server-side processing/CPU-intensive operations
  (πχ δηλαδή ανάλυση εικόνας, μετατροπή αρχείων)

- Γιατί…
- Το Node.js είναι ένα περιβάλλον εκτέλεσης που εκτελεί JavaScript στην πλευρά του διακομιστή

- Όντας, αρχικά, μια front-end γλώσσα προγραμματισμού, η JavaScript χρησιμοποιεί ένα μόνο νήμα για την γρήγορη επεξεργασία εργασιών (single threaded)

- Το Node.js, θεωρείται single threaded καθώς επεξεργάζεται JavaScript, που είναι single threaded.

Full stack Javascript Tools & Frameworks



[Image source]

Install nodejs

- Go to -> https://nodejs.org/en/download/
- Select your operating system
- Select the long-term support version, as recommended
- Download and run installation

Test if installed

Open

- terminal (mac)

or

- Command prompt /powershell (windows)

And run the following command

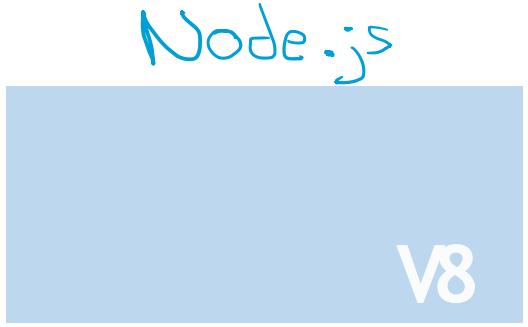
```
C:\Users\Aristea>node -v
v20.9.0
```

UNIVERSITY OF PIRAEUS

Node.js Modules

Node.js

- Είπαμε ότι το Node.js αποτελεί ένα **περιβάλλον εκτέλεσης JavaScript** βασισμένο στην ανοιχτού κώδικα v8 Javascript engine της Google.
  (asynchronous event-driven JavaScript runtime environment)



Node.js is like a container where javascript can be executed. Outside of any browser.

Node.js

- Nodejs είναι βέβαια κάτι πολύ παραπάνω από ένα «δοχείο» εκτέλεσης της V8 engine
- Το Nodejs μας δίνει περισσότερες δυνατότητες από αυτές που μας έδινε η javascript στον browser
- Ανάγνωση/δημιουργία εγγραφή σε αρχεία, σύνδεση σε βάση δεδομένων, δημιουργία server κλπ.
- Το Node.js μας παρέχει μια πληθώρα από JavaScript modules που απλοποιούν την ανάπτυξη εφαρμογών
- More about V8 engine: https://v8.dev/

- Έστω ότι έχουμε ένα αρχείο το

```
function a() {
   console.log("a");
}
a();
console.log('end');
```

1.Σε περιβάλλον περιηγητή με HTML:

- Συνήθως, JavaScript που προορίζεται για περιηγητή ενσωματώνεται σε ένα αρχείο HTML. Αυτό γίνεται περιλαμβάνοντας μια ετικέτα **<script>** στο HTML, η οποία είτε περιέχει άμεσα τον κώδικα JavaScript είτε αναφέρεται σε ένα εξωτερικό αρχείο JavaScript χρησιμοποιώντας το χαρακτηριστικό **src**. Για να εκτελέσετε αυτόν τον κώδικα, απλά ανοίγετε το αρχείο HTML σε έναν περιηγητή και ο JavaScript εκτελείται καθώς φορτώνεται η σελίδα.

- Έστω ότι έχουμε ένα αρχείο το

```
function a() {
  console.log("a");
}
a();
console.log('end');
```

Χρησιμοποιώντας Node.js στο τερματικό:

- Το Node.js είναι ένα περιβάλλον εκτέλεσης που σας επιτρέπει να εκτελείτε JavaScript στην πλευρά του διακομιστή ή σε ένα περιβάλλον γραμμής εντολών. Σε αυτό το πλαίσιο, δεν χρειάζεται αρχείο HTML. Αντ' αυτού, γράφετε τον κώδικα JavaScript σε ένα και το εκτελείτε εκτελώντας μια εντολή στο τερματικό: **node nameoffile.js**.

Ορισμένες Δυνατότητες

- Node.js επιτρέπει λειτουργικότητες που δεν είναι διαθέσιμες στο browser…

1.Πρόσβαση στο Σύστημα Αρχείων: Το Node.js παρέχει το module **fs**, το οποίο επιτρέπει την εκτέλεση λειτουργιών αρχείων όπως το διάβασμα, η εγγραφή, η διαγραφή, και η μετονομασία αρχείων στο τοπικό σύστημα αρχείων.

2.Λογική Πλευράς Διακομιστή: Με το Node.js, μπορείτε να χτίσετε web servers και να χειρίζεστε HTTP requests /responses

Node.js modules

- Το Node.js είναι κατασκευασμένο γύρω από την έννοια των Μονάδων (Modules).
- Οι Μονάδες στο Node.js μπορούν να θεωρηθούν ίδιες με τις βιβλιοθήκες JavaScript. Πρόκειται για ένα σύνολο λειτουργιών που θέλετε να συμπεριλάβετε στην εφαρμογή σας.
- Το Node.js περιλαμβάνει ένα σύνολο ενσωματωμένων μονάδων (δεν απαιτείται περαιτέρω εγκατάσταση).

Node.js modules

- include a module-**> require()** function + **name** of module

Examples:

Node.js includes an **HTTP module** that allows it to transfer data over the Hyper Text Transfer Protocol (HTTP is an application protocol, is the set of rules for transferring files -- such as text, images, sound, video, and other multimedia files over the web)

to **access the HTTP module** & create a server:

- var http = require('http’);

Node.js modules

- include a module-**> require()** function + **name** of module

fs module: μας επιτρέπει να αλληλοεπιδρούμε με το filesystem

- var filesystem=require('fs');

- Στη συνέχεια θα δούμε και την νεότερη σύνταξη με το import…

Node.js modules

Η μεταβλητή αποθηκεύει το αποτέλεσμα της μονάδας (module).

Έτσι, αποκτούμε πρόσβαση σε διάφορες λειτουργίες βάσει της χρησιμοποιούμενης μονάδας.

Στην πραγματικότητα, η συνάρτηση **require()** επιστρέφει ένα αντικείμενο που περιέχει πολλές μεθόδους στις οποίες μπορούμε εύκολα να έχουμε πρόσβαση.

Node.js modules ie.

```
JS read_write_fs.js > ...
1   const filesystem=require('fs');
2   // readFileSync takes as arguments the filepath and the character encoding
3   var textread= filesystem.readFileSync('./txt/filetoread.txt','utf8');
4
```

Η μέθοδος **fs.readFileSync()** χρησιμοποιείται για να:

- διαβάζει αρχεία και να επιστρέφει τα περιεχόμενά τους,
- διαβάζει αρχεία συγχρονισμένα, δίνοντας εντολή στο Node.js να σταματήσει όλες τις διεργασίες.

Έτσι, το αρχικό πρόγραμμα Node παύει ενώ περιμένει να ολοκληρωθεί η συνάρτηση **fs.readFileSync()**. Μόλις αυτό συμβεί, εκτελείται το υπόλοιπο πρόγραμμα Node.

Note that…

In general, files and directories maintain a tree structure for easy access.

There are 2 ways of getting the current directory in Node.js.

- **__dirname ->** returns the path of the folder where the **current JavaScript file resides**

```
JS read_write_fs.js > ...
1   const filesystem=require('fs');
2   // readFileSync takes as arguments the filepath and the character encoding
3   var textread= filesystem.readFileSync(__dirname+'/txt/filetoread.txt','utf8');
4
```

Note that…

In general, files and directories maintain a tree structure for easy access.

There are 2 ways of getting current directory in Node.js .

- **./ -> ** gives the **current working directory**. Current working directory is the path of the folder where the **node command** is executed and may change during the execution of the script.

- If for example we run node from the desktop, ./ will refer to the desktop.

Note that…

- The **only case** when ./ gives the **path of the currently executing file** is when it is used with the **require() command**

- Both __dirname and ./ give similar results when the node is running in the same directory as the currently executing file but produce different results when node.js run from some other directory.

Lets see an example

- Lets see how we can read and write files with node.js

Code example

Read a file

Create or overwrite file

JS read_write_fs.js ✕ | filetoread.txt

JS read_write_fs.js > ...

```
1   const filesystem=require('fs');
2   // readFileSync takes as arguments the filepath and the character encoding
3   var textread= filesystem.readFileSync(__dirname+'/txt/filetoread.txt','utf8');
4
5   //we create some text
6   var sometext= textread+'\n I will be your teacher for this course! Yeah!';
7
8   //we write that text to a file that is created once the script is executed
9   //if we use this function to write in a file it will overwrite the content
10  filesystem.writeFileSync('./txt/output.txt',sometext);
11
12  //read file created or overwiten
13  var newtextread= filesystem.readFileSync(`${__dirname}/txt/output.txt`,'utf8');
14  console.log(newtextread);
```

Create our own modules

- We may easily create and incorporate our own modules into our apps
- In order to do so we are going to use **module.exports**
- module.exports ->is used to export any var, function or object as a module

Create our own modules

- module.exports actually is a special object that is included in every JavaScript file in the Node.js application.
- It is an empty object by default, and its primary purpose is to define what **should be exported from a particular module**.
- When you create a module in Node.js, you can use module.exports to **expose** variables, functions, or objects from that module, making them accessible to other modules that require or import the module.

module.exports vs exports

module.exports:

- actual object returned when you require a module in another file.
- If you assign a new value to module.exports, it will **completely replace** the existing exports and become the **new exports object**.
- You can use module.exports to directly export a single object, function, or primitive value.

exports:

- It is a shorthand notation for module.exports.
- If you assign a new value to exports, it will **add properties to the existing module.exports object**.
- However, directly assign a new value to exports and expect it to replace the entire exports object.

UNIVERSITY OF PIRAEUS

To be continued…

https://nodejs.org/en/

https://www.w3schools.com/nodejs/nodejs_intro.asp

https://www.toptal.com/javascript/why-the-hell-would-i-use-node-js


To be continued...