Chapter 1

Getting started with Node.js

1

Outline

- A brief history of JavaScript and Node.js
- Overview of modern Javascript (ES6, ES7, and ES8). TypeScript
- The Node.js Ecosystem
- Node.js architecture
- Installing Node.js
- Running the first application: Hello World!

2

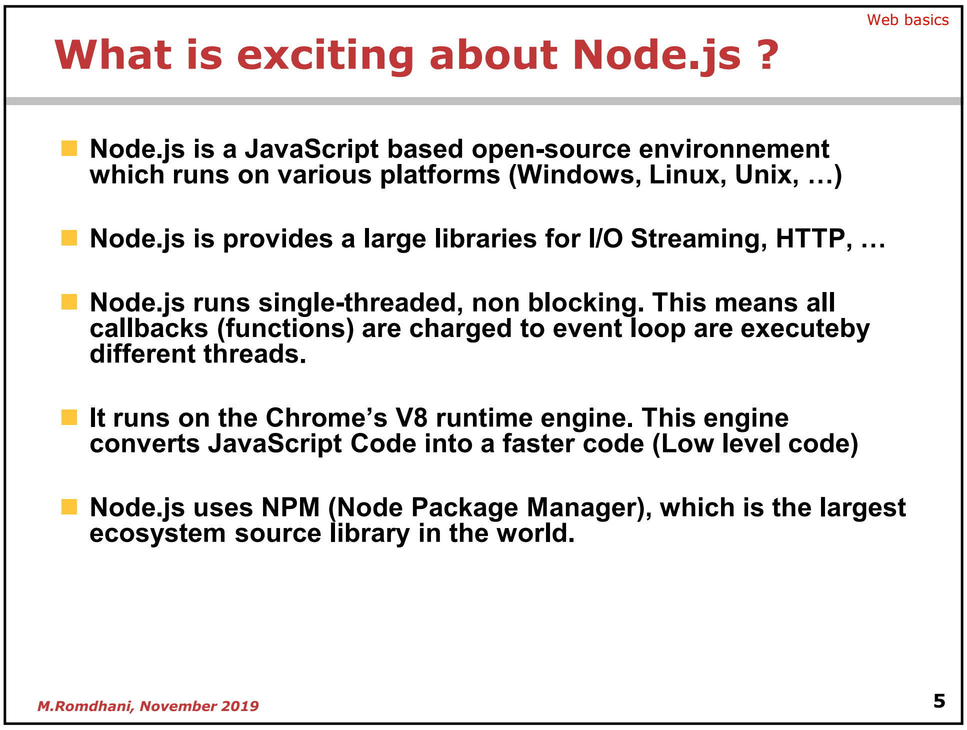
A brief history of JavaScript and Node.js

3

What is Node.js ?

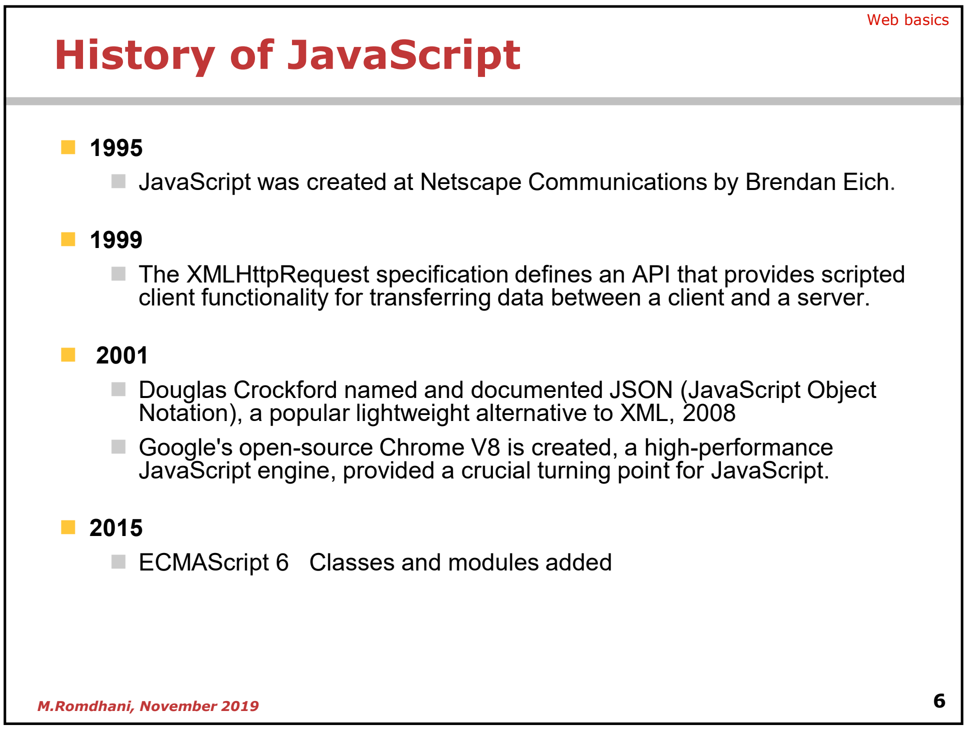
- Node.js is an open-source and cross-platform JavaScript runtime environment.
- Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.
- Node.js uses the JavaScript language which is used by millions of frontend developers that write for the browser. These are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language

4



What is exciting about Node.js ?

- **Node.js is a JavaScript based open-source environnement which runs on various platforms (Windows, Linux, Unix, …)**
- **Node.js is provides a large libraries for I/O Streaming, HTTP, …**
- **Node.js runs single-threaded, non blocking. This means all callbacks (functions) are charged to event loop are executeby different threads.**
- **It runs on the Chrome's V8 runtime engine. This engine converts JavaScript Code into a faster code (Low level code)**
- **Node.js uses NPM (Node Package Manager), which is the largest ecosystem source library in the world.**



History of JavaScript

- **1995**
  - JavaScript was created at Netscape Communications by Brendan Eich.
- **1999**
  - The XMLHttpRequest specification defines an API that provides scripted client functionality for transferring data between a client and a server.
- **2001**
  - Douglas Crockford named and documented JSON (JavaScript Object Notation), a popular lightweight alternative to XML, 2008
  - Google's open-source Chrome V8 is created, a high-performance JavaScript engine, provided a crucial turning point for JavaScript.
- **2015**
  - ECMAScript 6   Classes and modules added

A brief history of Node.js

- **Node.js is 12 years old! In comparison, JavaScript is 26 years old and the Internet is 32 years old.**
- **In 2009, Ryan Dahl wrote Node.js.**
  - On November 8, 2009 at the inaugural European JSConf, the Node.js project was first demonstrated by Dahl. Node.js is a combination the V8 JavaScript Chrome engine, a low-level I/O API, and an event loop.
- **Web frameworks were developed by the Node.js open-source community to accelerate the development of applications.**
  - These frameworks include Connect, Sails.js, Koa.js, Express.js, Feathers.js, socket.io, Derby, Hapi.js, Meteor, and a lot more.

A brief history of Node.js

- **2009**
  - The beginning of Node.js
  - npm was created
- **2010**
  - The beginning of Express
  - The beginning of socket.io
- **2011**
  - Version 1.0 of npm was released
  - Companies Uber, LinkedIn, etc. started adopting Node.js
  - The beginning of hapi
- **2012**
  - Adoption continues and growing rapidly

A brief history of Node.js

- **2018**
  - Node.js 10 (LTS)
  - ES modules .mjs experimental support
  - Node.js 11
- **2019**
  - Node.js 12 (LTS)
  - Node.js 13
- **2020**
  - Node.js 14 (LTS)
  - Node.js 15

After six months, odd-numbered releases (9, 11, etc.) become unsupported, and even-numbered releases (10, 12, etc.) move to *Active LTS* status and are ready for general use.

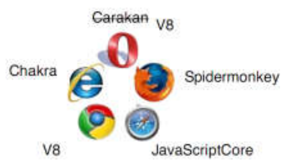
Overview of modern Javascript (ES6, ES7, and ES8) and TypeScript

What is JavaScript ?

- **A lightweight programming language ("scripting language")**
  - Client side processing
    - DOM Processing
    - Event Handling
    - Validation
    - Ajax
    - HTML 5 APIs
  - Server-side development
    - Web applications and mobile applications
    - Micro-services
    - Database Access
- **A web standard (but not supported identically by all browsers)**
  - JavaScript is an implementation of ECMAScript which conforms to the ECMAScript specification.
    - ECMAScript is a programming language itself.
    - The last version is ECMAScript 2020 (ES2020)

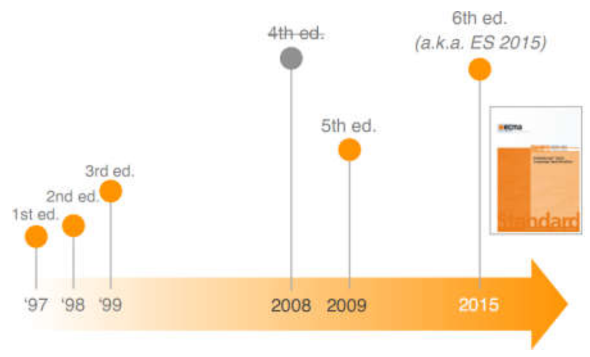
Overivew of JavaScript History

- **JavaScript's origins**
  - Invented by Brendan Eich in 1995, to support client-side scripting in Netscape Navigator
    - First called LiveScript, then JavaScript, then standardized as ECMAScript
    - Microsoft "copied" JavaScript in IE JScript
- **ECMAScript: "Standard" JavaScript**
  - The JavaScript "standardisation committee" has representatives from major Internet companies, browser vendors, web organisations, popular JS libraries and academia. Meets bi-monthly.
  - Current ECMAScript Engines



Overivew of JavaScript History

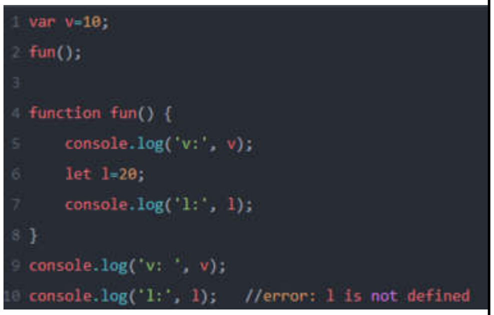
- **A Brief History of ECMAScript Specification**



  - ES 7 : Ecmascript 2016 (June 2016)
  - ES 8 : Ecmascript 2017 (June 2017)
  - ES 9 : Ecmascript 2018 (June 2018)
  - ES 10 : Ecmascript 2019 (June 2019)
  - ES 11: Ecmascript 2020 (June 2020)

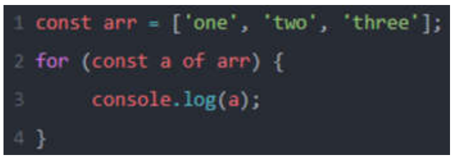
ECMAScript 6 new Features

- **let, const :**
  - Till ES5, JS has only function scope and global scope with the introduction of let keyword in ES6, JS can now have block scope.



```
var v=10;
fun();

function fun() {
    console.log('v:', v);
    let l=20;
    console.log('l:', l);
}
console.log('v: ', v);
console.log('l:', l);    //error: l is not defined
```

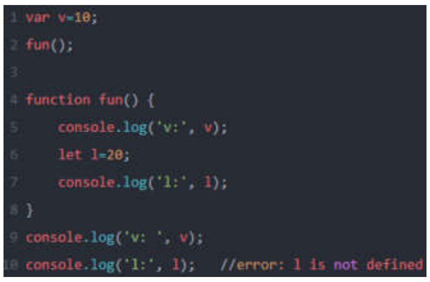
- **for .. Of**
  - for...of is an alternative for both for...in and forEach() and loops iterable data structures like Arrays, Maps, Sets, and strings



```
const arr = ['one', 'two', 'three'];
for (const a of arr) {
    console.log(a);
}
```

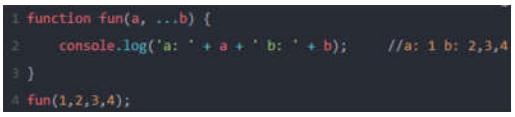
ECMAScript 6 new Features

- **Default parameter values:**
  - Provides default values to function parameters if no value or undefined is passed.



```
var v=10;
fun();

function fun() {
    console.log('v:', v);
    let l=20;
    console.log('l:', l);
}
console.log('v: ', v);
console.log('l:', l);   //error: l is not defined
```

- **Rest operator:**
  - Rest Operator is used to handle



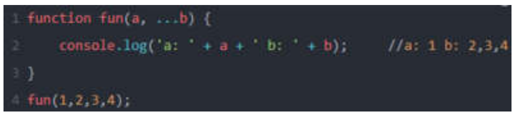
```
function fun(a, ...b) {
    console.log('a: ' + a + ' b: ' + b);     //a: 1 b: 2,3,4
}
fun(1,2,3,4);
```

ECMAScript 6 new Features

- **Default parameter values:**
  - Provides default values to function parameters if no value or undefined is passed.

```
var v=10;
fun();

function fun() {
    console.log('v:', v);
    let l=20;
    console.log('l:', l);
}
console.log('v: ', v);
console.log('l:', l);   //error: l is not defined
```

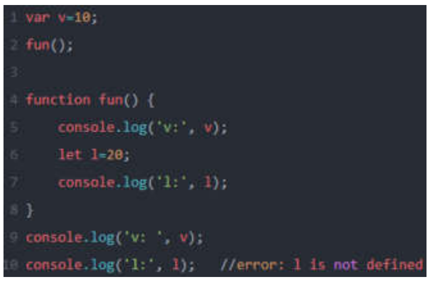
- **Rest operator:**
  - Rest Operator is used to handle



```
function fun(a, ...b) {
    console.log('a: ' + a + ' b: ' + b);     //a: 1 b: 2,3,4
}
fun(1,2,3,4);
```

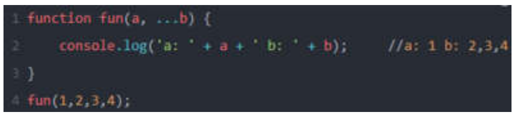
ECMAScript 6 new Features

- **Default parameter values:**
  - Provides default values to function parameters if no value or undefined is passed.



```
var v=10;
fun();

function fun() {
    console.log('v:', v);
    let l=20;
    console.log('l:', l);
}
console.log('v: ', v);
console.log('l:', l);   //error: l is not defined
```

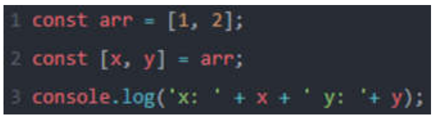
- **Rest operator:**
  - Rest Operator is used to handle



```
function fun(a, ...b) {
    console.log('a: ' + a + ' b: ' + b);    //a: 1 b: 2,3,4
}
fun(1,2,3,4);
```

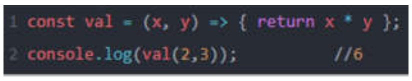
ECMAScript 6 new Features

- **Destructuring**
  - Destructuring helps in unpacking values from an array or an object.



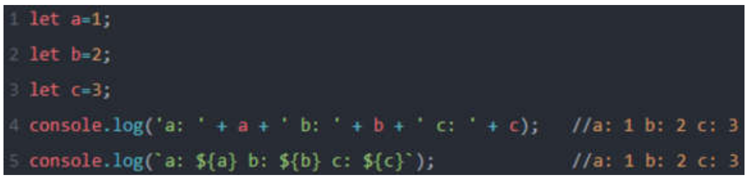
```
const arr = [1, 2];
const [x, y] = arr;
console.log('x: ' + x + ' y: '+ y);
```

- **ARROW FUNCTIONS:**
  - Arrow Functions use => as its token



```
const val = (x, y) => { return x * y };
console.log(val(2,3));          //6
```

- **Template strings:**
  - It allows embedded expressions, which makes print statements easy.

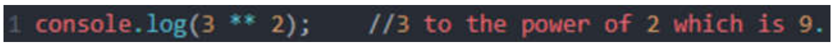


```
let a=1;
let b=2;
let c=3;
console.log('a: ' + a + ' b: ' + b + ' c: ' + c);    //a: 1 b: 2 c: 3
console.log(`a: ${a} b: ${b} c: ${c}`);              //a: 1 b: 2 c: 3
```

- **Other features : Classes, Promises, Modules, Proxies, …**

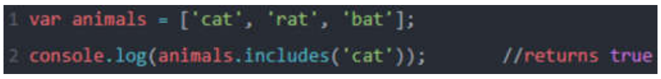
ECMAScript 7 new Features

- **Exponentiation operator**
  - ES7 added an exponentiation operator (**) to already JavaScript supported arithmetic operations like +,-,*. This operator raises the first operand to the power second operand.



```
console.log(3 ** 2);    //3 to the power of 2 which is 9.
```

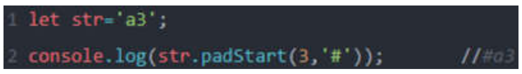
- **Includes**
  - Returns true if an array includes a value, if not returns false.Template strings:



```
var animals = ['cat', 'rat', 'bat'];
console.log(animals.includes('cat'));      //returns true
```

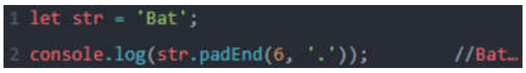
ECMAScript 8 new Features

- **padStart():**
  - This method pads a string with another string at the beginning.



```
let str='a3';
console.log(str.padStart(3,'#'));      //#a3
```

- **padEnd():**
  - This method pads a string with another string and makes the resulting string reach a given length. It adds spaces at the end of the string.



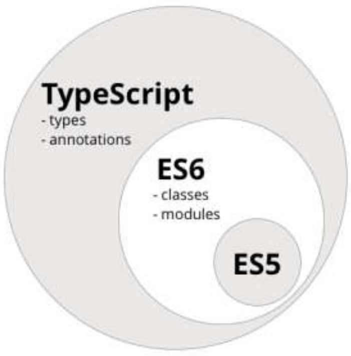
```
let str = 'Bat';
console.log(str.padEnd(6, '.'));      //Bat…
```

- **async/await**
  - Await operator, applied only inside an async function, waits to be rejected or resolved by a promise.

What is TypeScript?

- **Superset of JavaScript**
  - It was designed by Anders Hejlsberg (designer of C#) at Microsoft. The first version (0.8) was released in 2012. TypeScript is both a language and a set of tools
- **Compiles to plain JavaScript**
- **Strongly typed**
- **Class-based object-orientation**

.



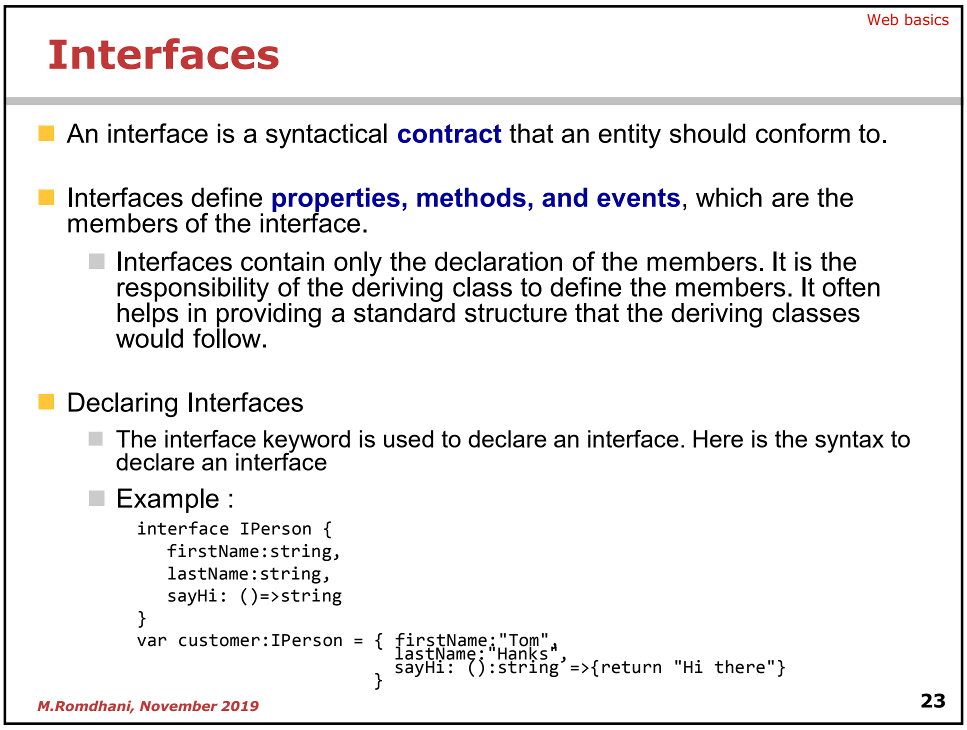
Strong Typing

JavaScript

```
let x;

x = 42;

let y = ‘Hello’;

let user =
    { id: 1111, name: ‘guest’ };
```

TypeScript

```
let x : number;
x = 42;

let y : string = ‘Hello’;

interface User {
   id: number;
   name: string
};

let user: User =
    { id: 1111, name: ‘guest’ };
```

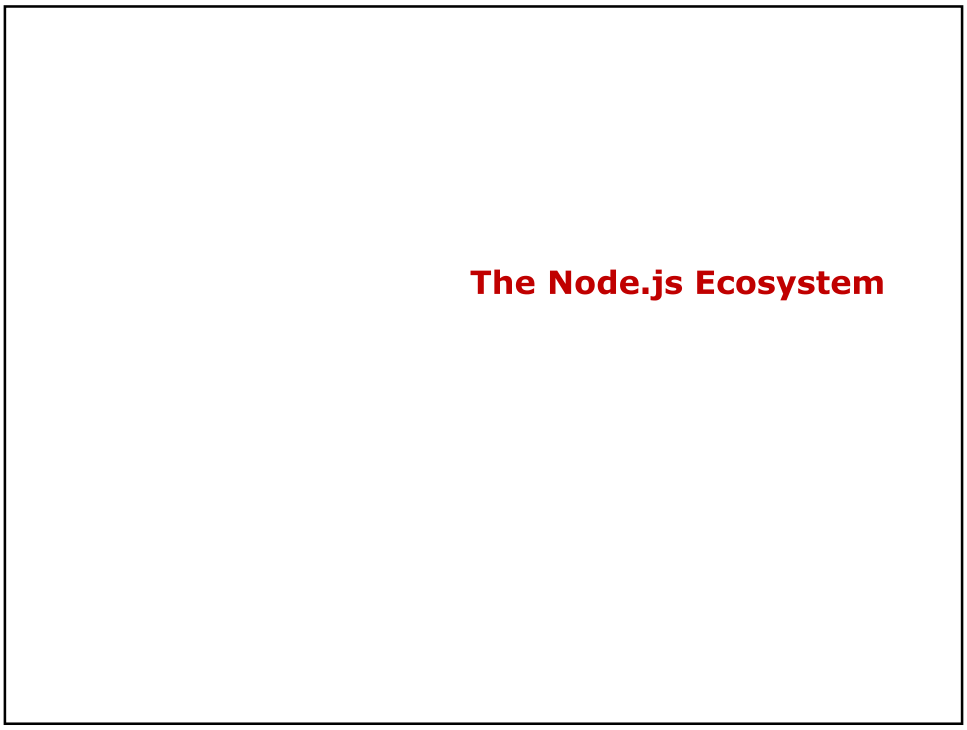


Interfaces

- An interface is a syntactical **contract** that an entity should conform to.
- Interfaces define **properties, methods, and events**, which are the members of the interface.
  - Interfaces contain only the declaration of the members. It is the responsibility of the deriving class to define the members. It often helps in providing a standard structure that the deriving classes would follow.
- Declaring Interfaces
  - The interface keyword is used to declare an interface. Here is the syntax to declare an interface
  - Example :

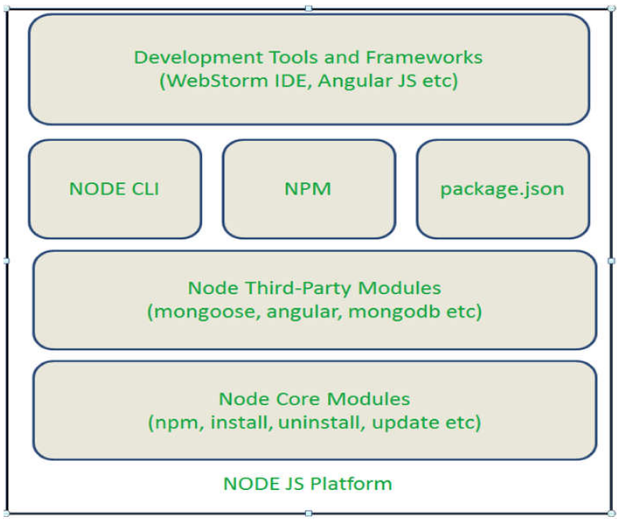
```
interface IPerson {
   firstName:string,
   lastName:string,
   sayHi: ()=>string
}
var customer:IPerson = { firstName:"Tom",
                         lastName:"Hanks",
                         sayHi: ():string =>{return "Hi there"}
                       }
```

23


The Node.js Ecosystem

24

NodeJS Platform


Development Tools and Frameworks
(WebStorm IDE, Angular JS etc)
NODE CLI
NPM
package.json
Node Third-Party Modules
(mongoose, angular, mongodb etc)
Node Core Modules
(npm, install, uninstall, update etc)
NODE JS Platform

Node.js Modules

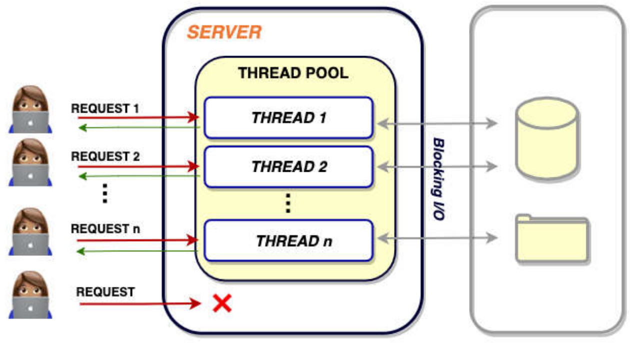
- **The node.js Modules**
  - Module in Node.js is a simple or complex functionality organized in single or multiple JavaScript files which can be reused throughout the Node.js application.
- **Types of modules**
  - Core modules
    - The core modules include bare minimum functionalities of Node.js. These core modules are compiled into its binary distribution and load automatically when Node.js process starts.
  - Third-party modules
    - Additional Libraries and frameworks for web development, data access, etc.

Node.js NPM

- **NPM is a package manager for Node.js packages, or modules if you like.**
  - The NPM program is installed on your computer when you install Node.js
- **www.npmjs.com hosts thousands of free packages to download and use.**
- **NPM is two things:**
  1. is an online repository for the publishing of open-source Node.js projects
  2. It is a command-line utility for interacting with said repository that aids in package installation, version management, and dependency management.

Node.js architecture

Traditional Multi-Threaded Processing Model in Web Frameworks

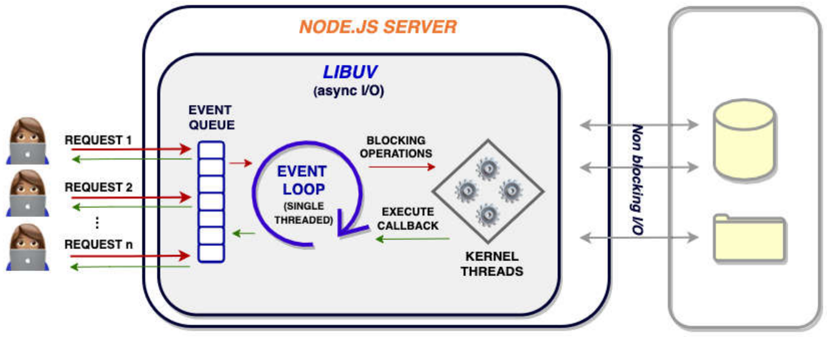


- If the threadpool is exhausted , the server is forced to wait for at least one of the busy threads to be freed for the new request(s) to be catered to.
- The synchronous nature of processing inside each thread means that even though we can spin up multiple threads for concurrent requests, each thread, individually, will be slowed down

M.Romdhani, November 2019 **29**

29

Single-Threaded Event Loop Architecture in Node.js



- The event loop can take care of basic processing itself, but for async I/O operations, involving modules such as fs (I/O-heavy) and crypto (CPU-heavy), it can offload the processing to the worker pool in the system kernel.
- While these threads work on their assigned operations, the event loop can continue operating as usual, concurrently catering to other requests.

M.Romdhani, November 2019 **30**

30

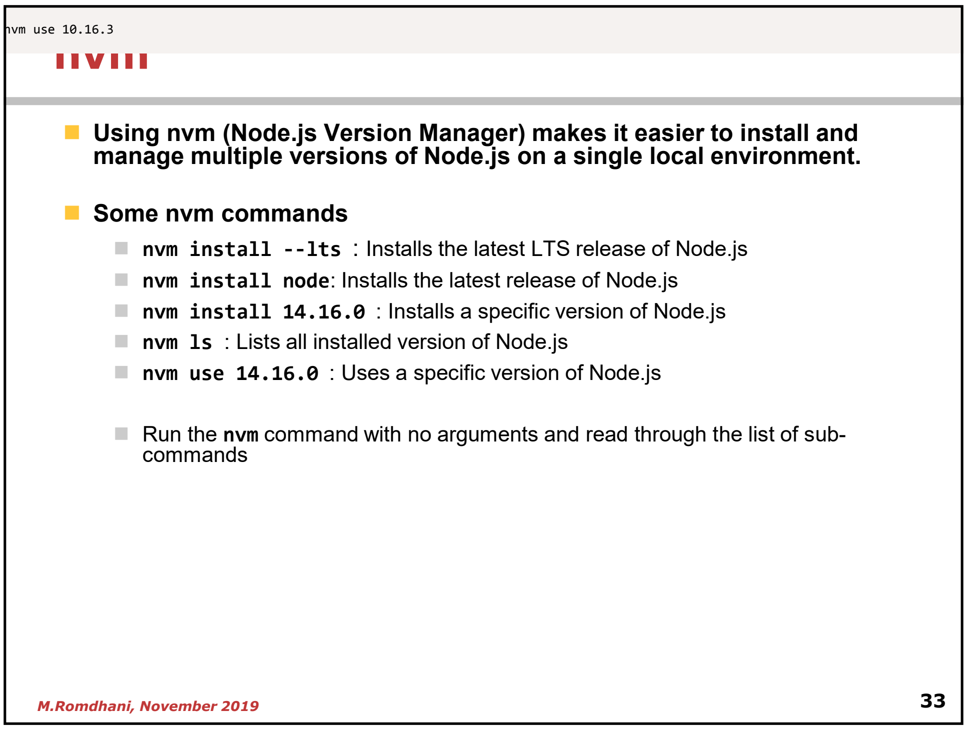
Installing Nodejs

31

Installing Node.js

- **Installation on Windows**
  - Nodejs Installer to install Node.js and npm
    - Installer available on https://nodejs.org/en/download/
  - Install using Chocolatey
    - choco install nodejs
  - Install using nvm
    - nvm install 14.16.0
- **Installation on MacOS**
  - Nodejs Installer to install Node.js and npm
    - Installer available on https://nodejs.org/en/download/
  - Install using nvm
- **Installation on Linux/Unix**
  - Use the Platform Package manager to install Node.js and npm
    - Apt-get, rpm, yum, …
  - Install using nvm

32



```
nvm use 10.16.3
```

nvm

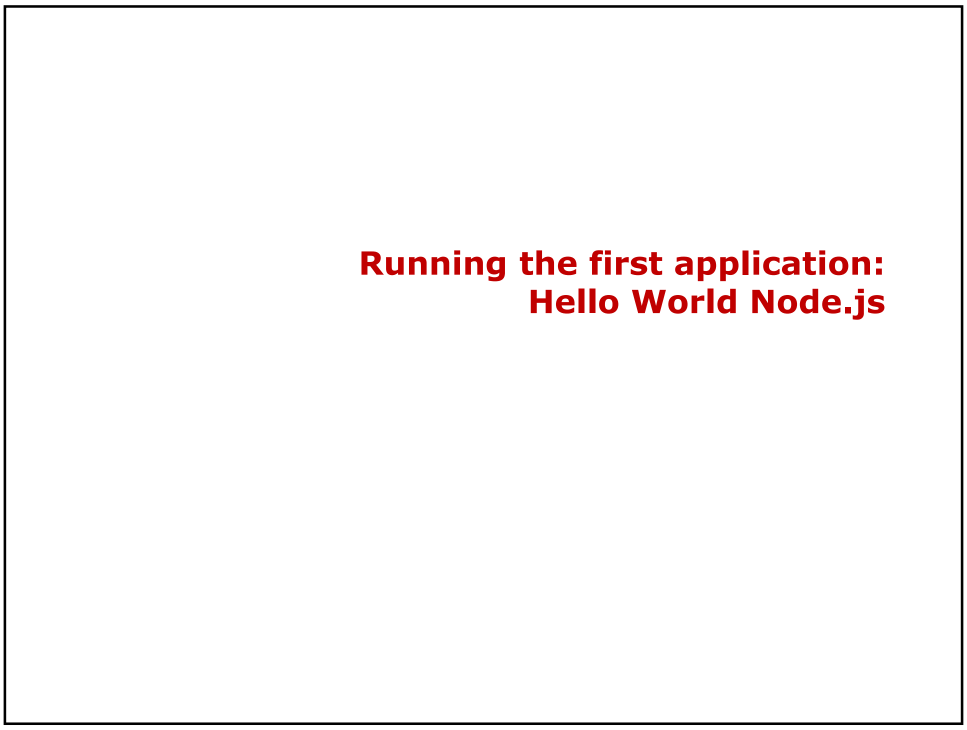
- **Using nvm (Node.js Version Manager) makes it easier to install and manage multiple versions of Node.js on a single local environment.**
- **Some nvm commands**
  - **`nvm install --lts`** : Installs the latest LTS release of Node.js
  - **`nvm install node`**: Installs the latest release of Node.js
  - **`nvm install 14.16.0`** : Installs a specific version of Node.js
  - **`nvm ls`** : Lists all installed version of Node.js
  - **`nvm use 14.16.0`** : Uses a specific version of Node.js

  - Run the **`nvm`** command with no arguments and read through the list of sub-commands

M.Romdhani, November 2019

33

33



Running the first application: Hello World Node.js

34

Using the Node.js REPL

- **REPL also known as Read Evaluate Print Loop is a programming language environment(Basically a console window) that takes single expression as user input and returns the result back to the console after execution**
- **The REPL mode is activated just by calling the `node` command.**
  - REPL will wait for JavaScript expressions and Javascript code
  - REPL provides autocomplete (with TAB key)
- **To exit REPL :**
  - .exit command or CTRL-C twice

Hello World, Node.js

- **Display simple message**

```
let message = 'Hello World';
console.log(message);

for (let index = 0; index < 10; index++) {
    console.log(`${index} : message`);
}
```

- **Simple Web Server**

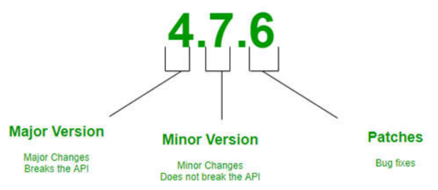
```
const http = require('http');

http.createServer(function (req, res) {
      res.writeHead(200, {'Content-Type': 'text/html'});
      res.end('Hello World!');
 }).listen(3000,
       ()=> console.log('The server is listening on port 3000'));
```

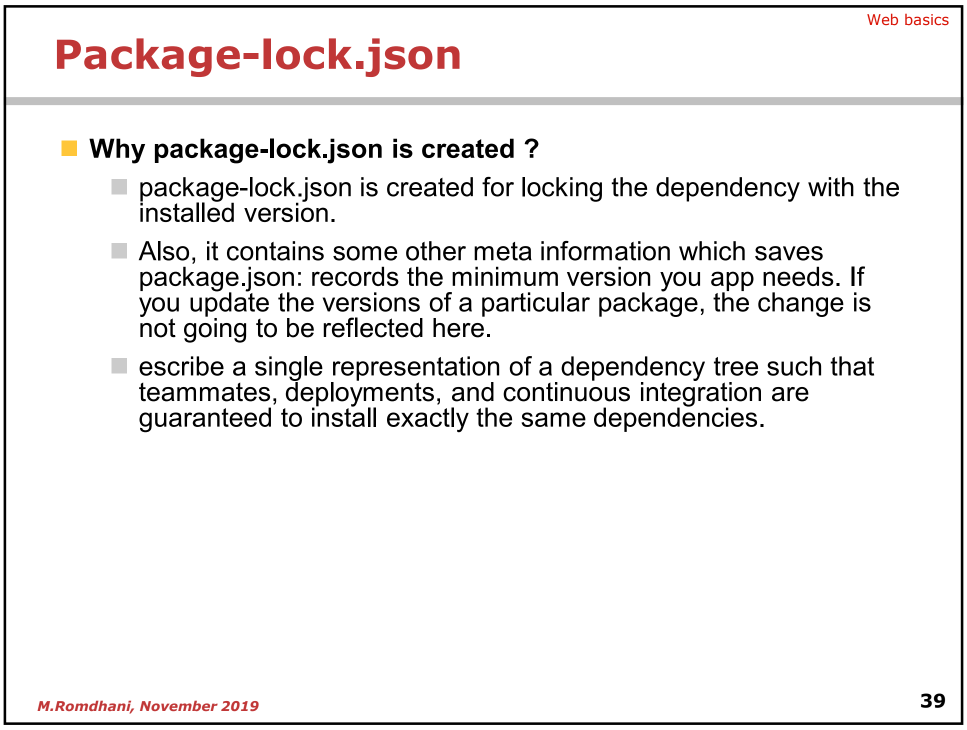
npm

- **npm (originally short for Node Package Manager) is the package manager for the JavaScript programming language.**
- **It consists of:**
  - a command line client, also called npm, and
  - an online database of public and paid-for private packages, called the npm registry.
- **Some npm commands :**
  - `npm init` : Creates a package.json in the project folder
  - `npm install --global @typescript` : Installs globally the typescript compiler
  - `npm install --save bootstrap` : Installs Bootstrap locally in the project
  - `npm install -save-dev mocha` : Installs the latest stable version of Bootstrap
  - `npm update axios` : updates the axios package
  - `npm unistall axios` : uninstalls the axios package

package.json

- **`package.json` file can be described as a manifest of your project that includes the packages and applications it depends on, information about its unique source control, and specific metadata like the project's name, description, and author.**
  - There are two kinds of dependencies :
    - **`dev-dependencies`** : which are needed for quality control and production
    - **`dependencies`** that are needed for development
- **Semantic versioning**
  - Semantic versioning is a formal convention for specifying compatibility using a three-part version number: major version; minor version; and patch.
  - ~version “Approximately equivalent to version”,
    - ~1.2.3 will use releases 1.2.*
  - ^version “Compatible with version”,
    - ^2.3.4 will use releases from 2.*.*


4.7.6
Major Version
Major Changes Breaks the API
Minor Version
Minor Changes Does not break the API
Patches
Bug fixes



Package-lock.json

- **Why package-lock.json is created ?**
  - package-lock.json is created for locking the dependency with the installed version.
  - Also, it contains some other meta information which saves package.json: records the minimum version you app needs. If you update the versions of a particular package, the change is not going to be reflected here.
  - escribe a single representation of a dependency tree such that teammates, deployments, and continuous integration are guaranteed to install exactly the same dependencies.