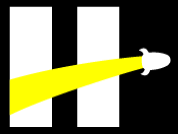

HENRY

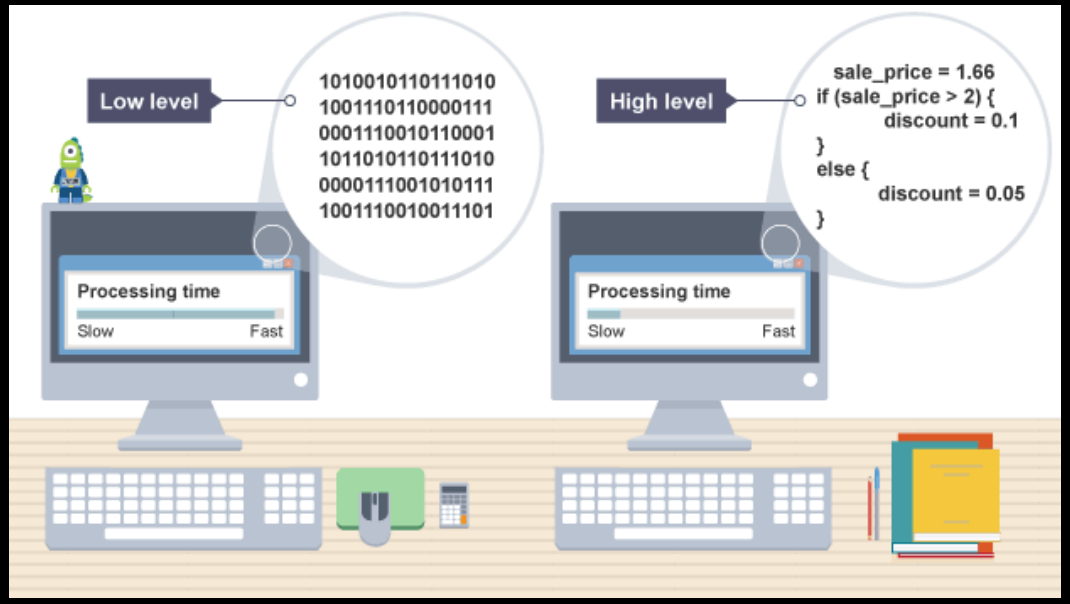


Node

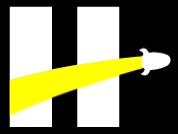
Lenguajes de Bajo Nivel y de Alto Nivel

```
-u 100 1a
0CFD:0100 BA0B01      MOV     DX,010B
0CFD:0103 B409        MOV     AH,09
0CFD:0105 CD21        INT     21
0CFD:0107 B400        MOV     AH,00
0CFD:0109 CD21        INT     21
-d 10b 13f
0CFD:0100                                               48 6F 6C 61 2C
0CFD:0110 20 65 73 74 65 20 65 73-20 75 6E 20 70 72 6F 67
0CFD:0120 72 61 6D 61 20 68 65 63-68 6F 20 65 6E 20 61 73
0CFD:0130 73 65 6D 62 6C 65 72 20-70 61 72 61 20 6C 61 20
0CFD:0140 57 69 6B 69 70 65 64 69-61 24
                                                    Hola,
 este es un prog
rama hecho en as
sembler para la
Wikipedia$
```

Assembler



Node está escrito en **C++**!



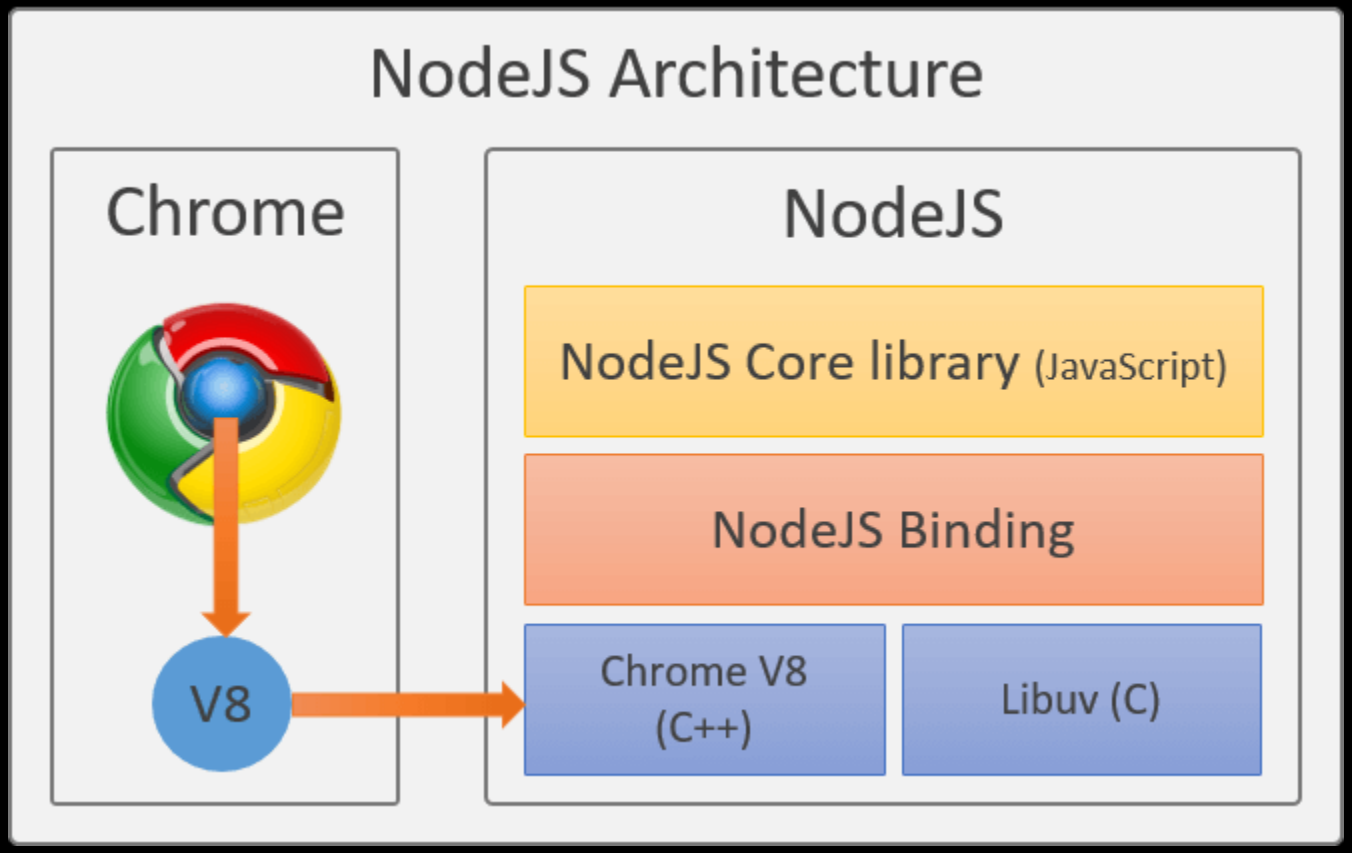
V8

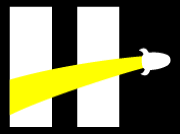
- V8 JavaScript Engine: *ya sabiamos*.
- V8 is Google's open source JavaScript engine: *really?*.
- V8 implements ECMAScript as specified in ECMA-262: *ah!, sigue estándares, bien!.*
- V8 is written in C++ and is used in Google Chrome, the open source browser from Google: *Alguien conoce ese browser Chrome, es bueno?*.
- V8 can run standalone, or can be embedded into any C++ application: *Atención con esto, se puede embeber (usar) en cualquier aplicación C++*

V8 + libuv

- Maneras de organizar nuestro código para que sea reusable (**módulos**)
- Poder leer y escribir archivos (input/output)
- Leer y escribir en Bases de Datos.
- Poder enviar y recibir datos de internet.
- Poder interpretar los formatos estándares.
- Alguna forma de manejar procesos que lleven mucho tiempo.

NodeJS

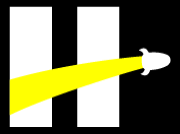

NodeJS Architecture
Chrome
V8
NodeJS
NodeJS Core library (JavaScript)
NodeJS Binding
Chrome V8 (C++)
Libuv (C)



NodeJS

```
// ejecutando el archivo

$ node index.js
$ 1
```

```
// index.js

const hola = 1;

console.log(hola);
```



Módulos

Def: Un bloque de código reusable, cuya existencia no altera accidentalmente el comportamiento de otros bloques de código.

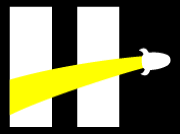
Cómo funciona CommonJs Modules?

Básicamente, el standart dice lo siguiente:

- Cada archivo es un ḿodulo, y cada módulo es un archivo separado.
- Todo lo que queremos exportar va a ser expuesto desde un único punto.

Require con modulos Core o nativos

```
1 var util = require('util'); // No usamos ./ porque es un modulo core
2
3 var nombre = 'Toni';
4 var saludo = util.format('Hola, %s', nombre);
5 util.log(saludo);
```



Gestor de Paquetes

Primero definamos lo que es un paquete. Básicamente es.. `código` (un módulo podría ser un paquete)! Es cualquier pieza de código manejada y mantenida por un gestor de paquetes.

Ahora, un gestor de paquetes es un software que automatiza la instalación y actualización de paquetes.

NPM: Node Package Manager

Package.json

Para poder trackear las dependencias y los paquetes que tenemos instalados (entre otras cosas), npm hace uso de un archivo de **configuración** al que llama **package.json**.

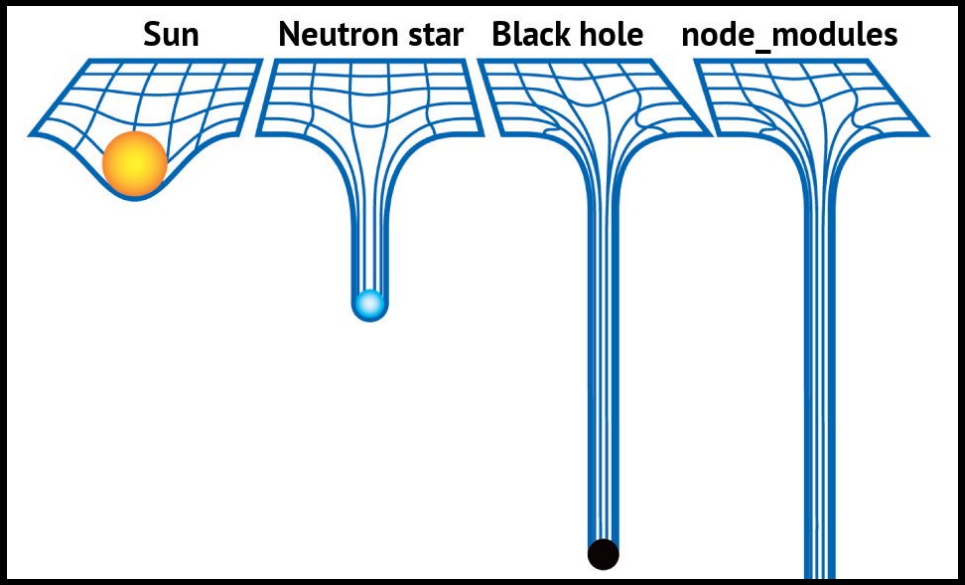
```json
{
  "main": "node_modules/expo/AppEntry.js",
  "scripts": {
    "start": "expo start",
    "android": "expo start --android",
    "ios": "expo start --ios",
    "web": "expo start --web",
    "eject": "expo eject",
    "test": "jest"
  },
  "dependencies": {
    "expo": "^35.0.0",
    "jest": "^24.9.0",
    "react": "16.8.3",
    "react-dom": "16.8.3",
    "react-native": "https://github.com/expo/react-native/archive/s
    "react-native-web": "^0.11.7"
  },
  "devDependencies": {
    "babel-preset-expo": "^7.0.0"
  },
  "private": true
}
```

```
// Para iniciar una carpeta
// con un package.json podemos hacer

npm init
```

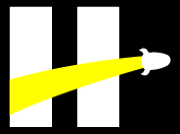
Semantic Versioning


```
1  ~1.2.3  is >=1.2.3 <1.3.0
2
3  ^1.2.3  is >=1.2.3 <2.0.0
```

NPM

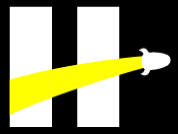


Los paquetes instalados de forma local serán guardados en una carpeta llamada `**node_modules**` creada dentro de la carpeta donde ejecuté el comando



NPM

```
// Algunos comandos

npm install --save {nombrePaquete}

npm install -g {nombre paquete}

npm update

npm audit

npm start

npm test

npm run {nombreScript}
```



< DEMO />