

**HENRY**

A bright yellow beam of light originates from the left edge of the frame and points towards the letter 'R' in the word 'HENRY'. The beam is wider on the left and tapers as it moves to the right. At the tip of the beam, where it meets the 'R', there is a small white icon of a rocket or missile head.



# Node

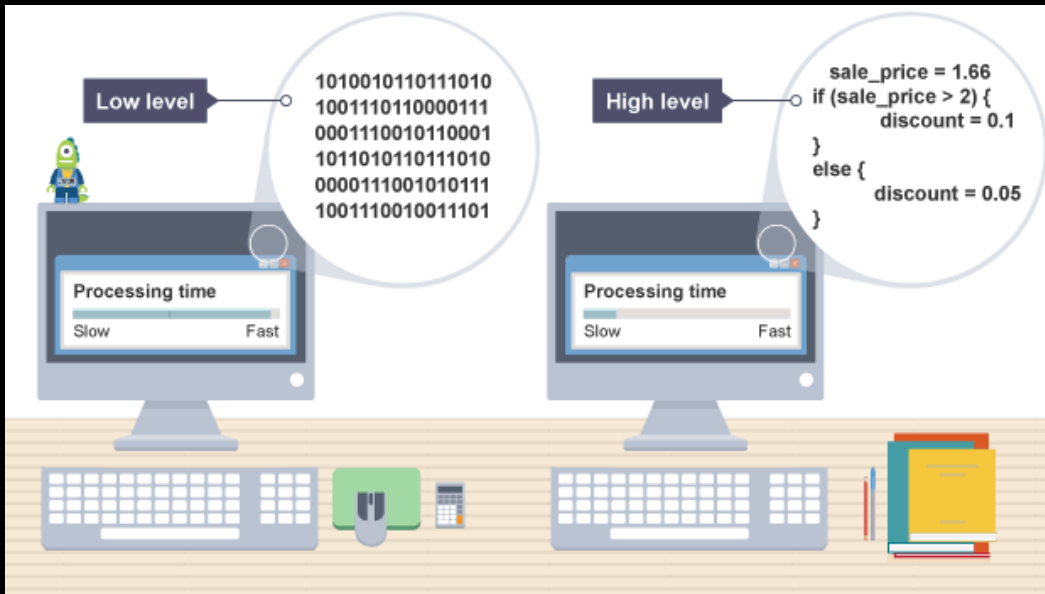


# Lenguajes de Bajo Nivel y de Alto Nivel



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

Assembler



Node está  
escrito en C++!



# V8



- V8 JavaScript Engine: *ya sabemos.*
- 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



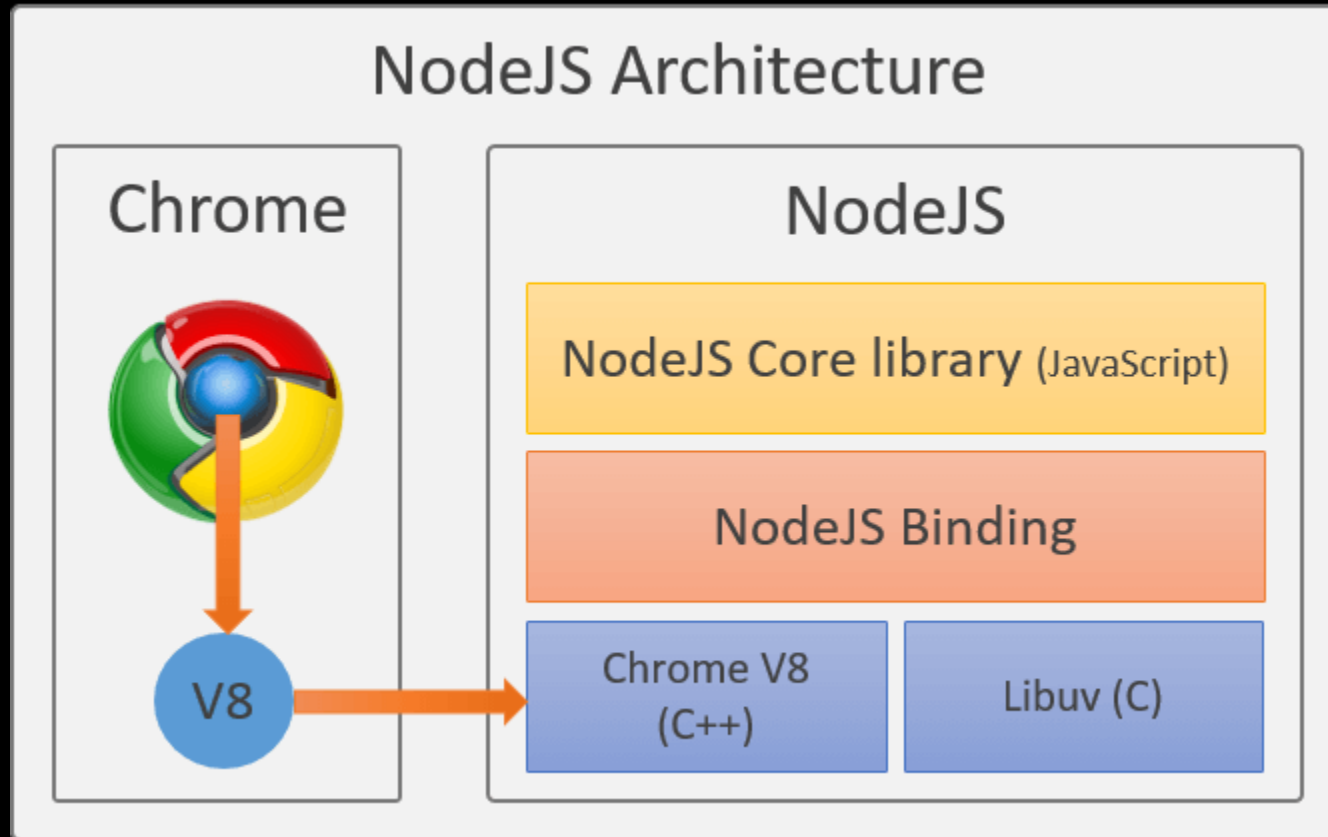
- Maneras de organizar nuestro código para que sea reusable (módulos)
- Poder leer y escribir archivos ( input/output)
- Leer y escribir en Bases de Datos.
- Poder enviar y recibir datos de internet.
- Poder interpretar los formatos estándares.
- Alguna forma de manejar procesos que lleven mucho tiempo.



# NodeJS



## NodeJS Architecture





# NodeJS



```
1 // ejecutando el archivo
2
3 $ node index.js
4 $ 1
```



```
1 // index.js
2
3 const hola = 1;
4
5 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 módulo, 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**.

```
{
  "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
}
```



```
1 // Para iniciar una carpeta
2 // con un package.json podemos hacer
3
4 npm init
```



# Semantic Versioning



1 . 3 . 1

MAJOR . MINOR . PATCH

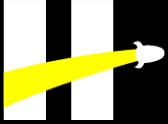
Breaking  
Changes

Nueva  
funcionalidad,  
retro-compatible

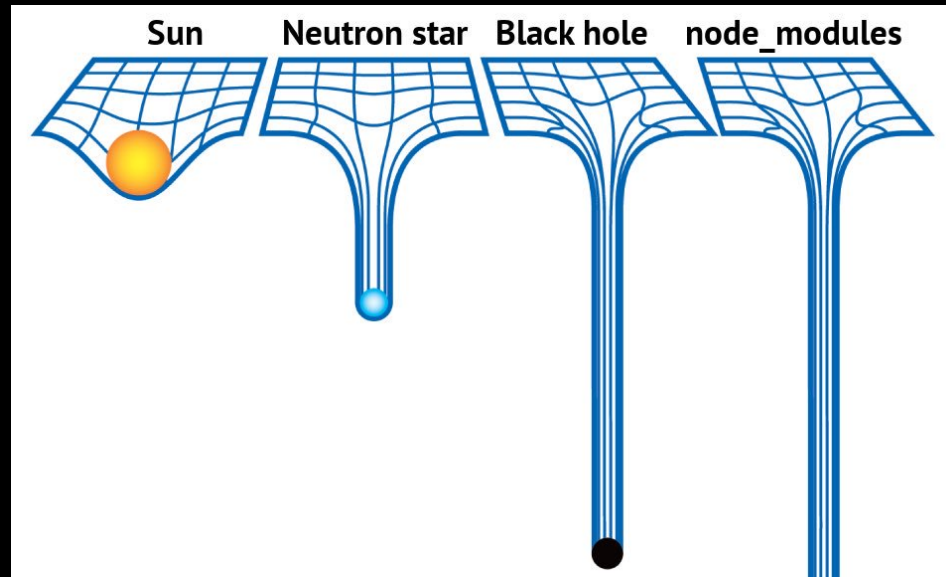
Bug fix  
retro-compatible



```
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



```
1 // Algunos comandos
2
3 npm install --save {nombrePaquete}
4
5 npm install -g {nombre paquete}
6
7 npm update
8
9 npm audit
10
11 npm start
12
13 npm test
14
15 npm run {nombreScript}
```



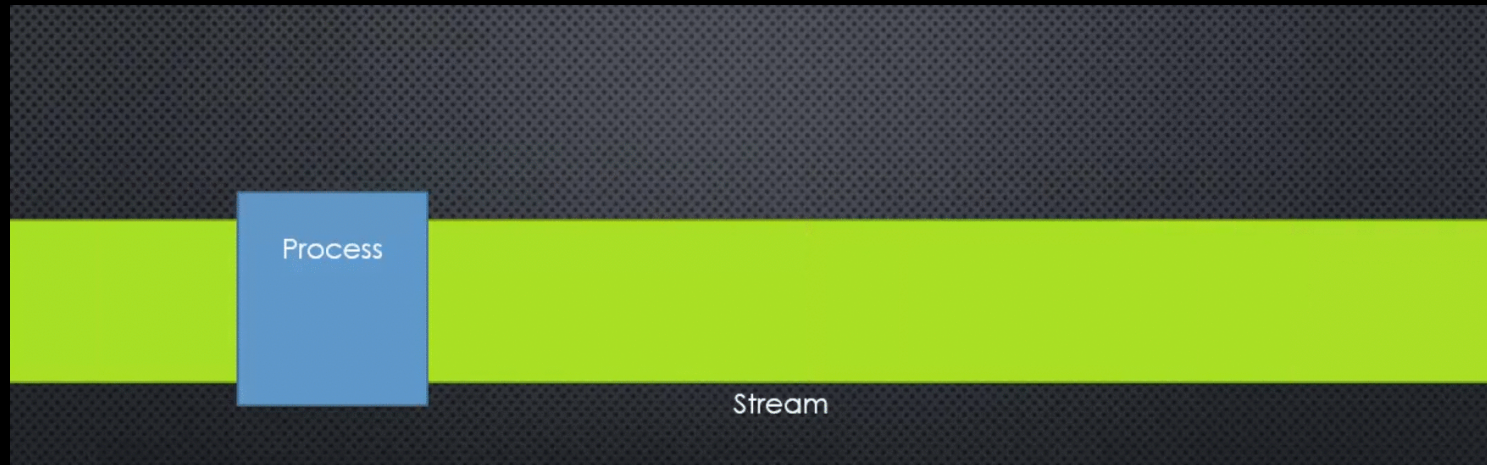
< DEMO />



# Streams, Buffers, y Pipes



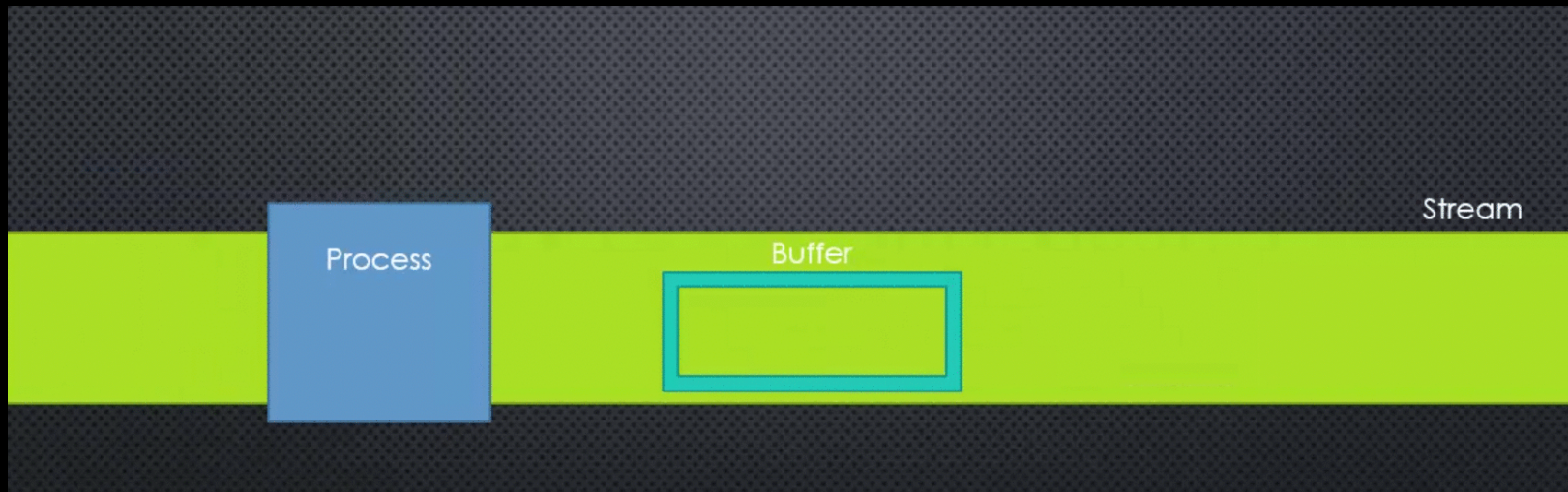
# Streams







# Buffers





# Pipes

