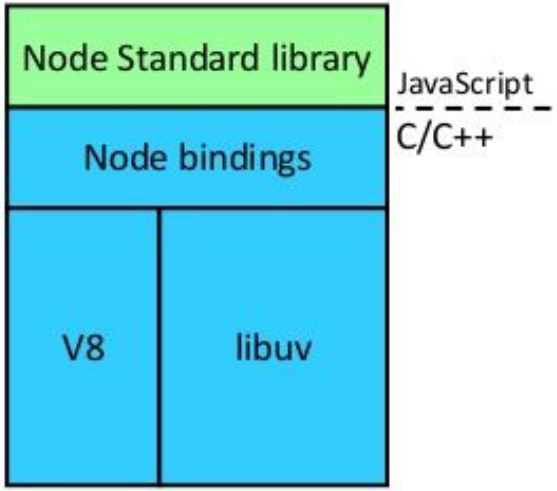
Node JS

Node JS: Qué es?

Node.js® es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente.

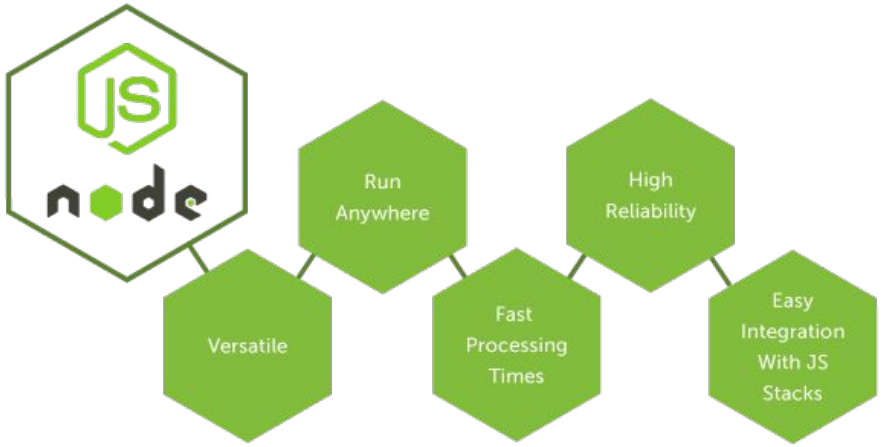
Overview of Node.js

- Node.js architecture



Node JS: Características

- Asíncrono y basado en eventos
- Muy rápido
- Un solo hilo altamente escalable
- No usa buffer
- Licencia: MIT



Node JS: Donde usar nodejs?

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- APIs basadas en JSON
- Single Page Applications

No usar en aplicaciones que hacen uso intensivo de CPU.

Node JS: Quienes lo usan?

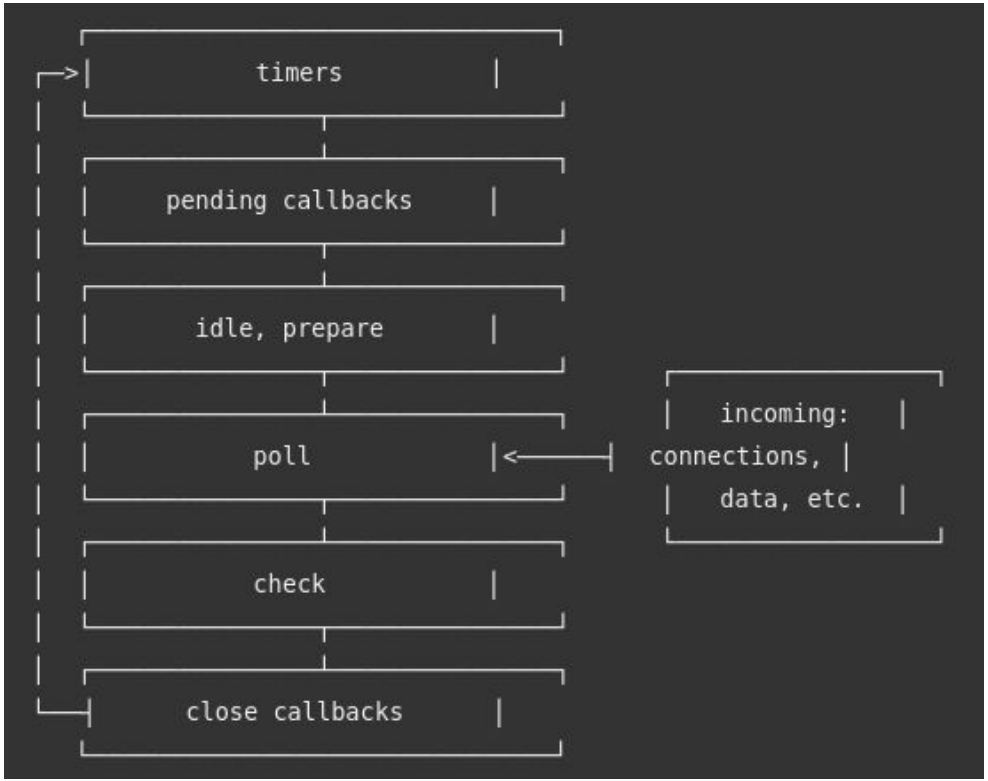
- PayPal
  - 2x velocidad de desarrollo
  - 33% menos código
- Netflix
  - 70% mejora en el tiempo de descarga
- Uber
  - 2M RPC por segundo
- Linkedin:
  - Reducción de servidores de 15 a 4
  - 2x capacidad de tráfico
  - 2x a 10x velocidad del lado del cliente

- Ebay
- Walmart
- Medium
- NASA
- Mozilla
- Trello

Node JS: Instalación

Descargar el instalador desde: https://nodejs.org/en/download/

NodeJS: Loop


timers
pending callbacks
idle, prepare
poll
check
close callbacks
incoming:
connections,
data, etc.

Node JS: Ejemplos

- Hola Mundo
- Timers
- nextTick
- Events

Node JS: Ejemplos

```
const fs = require('fs');
const data = fs.readFileSync('/file.md');
console.log(data);
```

```
const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

NodeJS: Ejemplos

```
const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
  if (err) throw err;
  console.log(data);
});
fs.unlinkSync('/file.md');
```

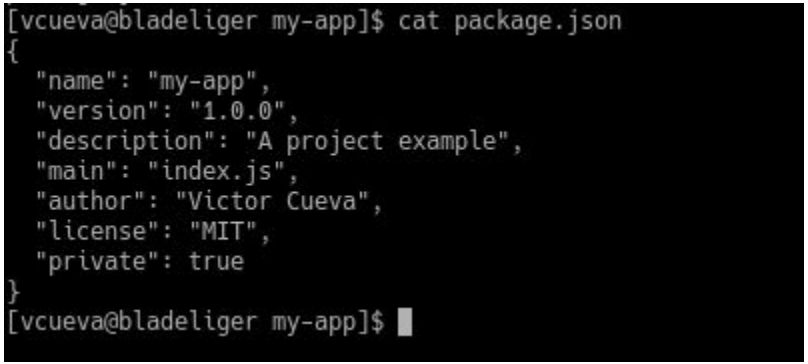
```
const fs = require('fs');
fs.readFile('/file.md', (readFileErr, data) => {
  if (readFileErr) throw readFileErr;
  console.log(data);
  fs.unlink('/file.md', (unlinkErr) => {
    if (unlinkErr) throw unlinkErr;
  });
});
```

Node JS: Gestor de paquetes

- npm
- yarn

Node JS: Proyecto

- npm init
- yarn init



Node JS: Instalación de dependencias

Dependencias principales

- npm install --save **<dependencia>**
- yarn add **<dependencia>**

Dependencias de desarrollo

- npm install --save-dev **<dependencia>**
- yarn add --dev **<dependencia>**

Node JS: Dependencias globales

- npm install -g **<dependencia>**
- yarn global add **<dependencia>**

ES 6

ES 6: Strict Mode

El modo estricto tiene varios cambios en la semántica normal de JavaScript:

1. Elimina algunos errores silenciosos de JavaScript cambiandolos a que lancen errores.
2. Corrige errores que hacen difícil para los motores de JavaScript realizar optimizaciones: a veces, el código en modo estricto puede correr más rápido que un idéntico no es estricto.
3. Prohíbe cierta sintaxis que probablemente sean definidas en futuras versiones de ECMAScript.

ES 6: Variables

```
const URL = 'www.mydomain.com';
URL = 'whatever'; // ERROR!!
```

```
(function () {
    console.log(global); // undefined
    console.log(local); // undefined
    if (true) {
        var global = "I'm global";
        let local = "I'm only local";
    }
    console.log(global); // I'm global
    console.log(local); //undefined
})();
```

ES 6: Funciones

```
setInterval(function () {
    console.log('hi world');
}, 100);
```

```
setInterval(() => {
    console.log('hi world');
}, 100);
```

```
var vowels = ['a', 'e', 'i', 'o', 'u'];
vowels.forEach(function (value) {
    console.log('vowel :' + value);
});
```

```
var vowels = ['a', 'e', 'i', 'o', 'u'];
vowels.forEach(value => {
    console.log('vowel :' + value);
});
```

ES 6: Funciones

```
let sum = function (a, b) {
    return a + b;
}
```

```
let sum = (a, b) => a + b;
```

```
function hello(name = 'Victor') {
    console.log('Hello ' + name);
}

hello();
hello('Josue');
```

ES 6: Funciones

```
function printName(name) {
    var length = arguments.length;
    var fullName = name;
    if (length > 1) {
        for (var i = 1; i < length; i++) {
            fullName += ' ' + arguments[i];
        }
    }
    console.log(fullName);
};

printName('Felipe');
printName('Felipe', 'Juan', 'Froilan');
```

```
function printName(name, ...fancyNames) {
    var fullName = name;
    fancyNames.forEach(fancyN => fullName += ' ' + fancyN);
    console.log(fullName);
};
printName('Felipe'); // Felipe
printName('Felipe', 'Juan', 'Froilan'); //Felipe Juan Froilan
```

ES: Clases

```
// Creacion
class Document {
    constructor(title, author, isPublished) {
        this.title = title;
        this.author = author;
        this.isPublished = isPublished;
    }
    publish() {
        this.isPublished = true;
    }
}

// Herencia
class Book extends Document {
    constructor(title, author, topic) {
        super(title, author, true);
        this.topic = topic;
    }
}
```

ES 6: Descomposicion

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

function foo() {
    return [175, 75];
};
let [height, weight] = foo();
console.log('hw:', height, weight);

let obj = {a: 5, b: 10};
let {a, b} = obj;
console.log('a*b:', a*b);

let [m, n, ...iterObj] = [1, 2, 3, 4, 5];
console.log(m, n, iterObj);
```

ES 6: Loops

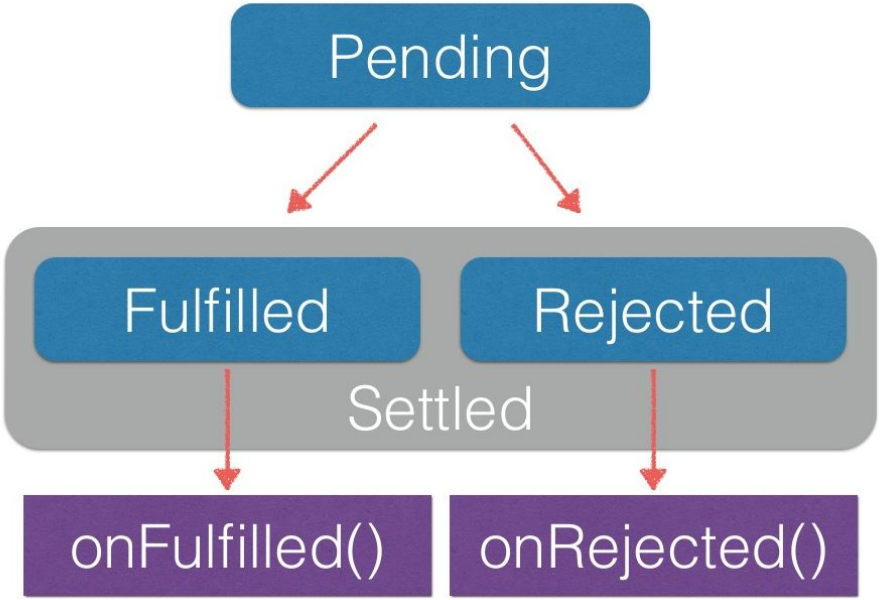
```
var numbers = [1, 2, 3, 4, 5];
for (let item of numbers) {
    console.log(item);
}
```

ES 6: Map y Set

```
let map = new Map();
map.set('foo', 123);

let user = { userId: 1 };
map.set(user, 'Alex');
map.get('foo'); //123
map.get(user); //Alex

for (let [key, value] of map) {
    console.log(key, value);
}

map.size; //2
map.has('foo'); //true
map.delete('foo'); //true
map.has('foo'); //false

map.clear();
map.size; //0
```

```
let set = new Set();
set.add('foo');
set.add('bar');
set.size //2
for (let item of set) {
    console.log(item);
}

set.has('foo'); //true
set.delete('foo'); //true
set.has('foo'); //false
set.size //1
set.clear();
set.size //0
```

ES 6: Promesas



ES 6: Promesas

Sintaxis:

```
new Promise(function(resolve, reject) { ... });
```

```
promise.then(
    function (value) {
        console.log("success");
    },
    function (reason) {
        console.log("error ", reason);
    }
);
```

Express JS

Express JS: Que es?

Web Oficial: http://expressjs.com/es/

Express JS: Instalación

- npm install --save express
- yarn add express

Express JS: Primera app

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

const app = express();

app.get((request, response) => {
    response.end('Hola Mundo');
});

const server = http.createServer(app);
let port = 3000;

server.listen(port, () => {
    console.log('Servidor corriendo en el puerto ' + port);
});
```

Express JS: Enrutamiento

- Rutas en el mismo archivo
- Rutas en archivos separados: Router

Express JS: Archivos estáticos

- express.static

Express JS: Parámetros en las rutas

- En el path: req.params
- En la query: req.query
- En el Body: req.body

Express JS: Middlewares

- Middlewares Útiles
  - Body parser
  - Cookie parser

- Middlewares globales
- Middlewares para rutas específicas

Express JS: cookies

- Leer las cookies
  - req.cookies

- Crear un cookie
  - res.cookie(clave, valor)

Express JS: Express-Generator

- npm install -g express-generator
- yarn global add express-generator

- express --git **<nombre del proyecto>**

Bibliografía

http://inubo.es/noticia/los-10-mejores-ejemplos-de-aplicaciones-node-js-para-empresas

https://nodejs.org/en/

http://expressjs.com/es/