FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Desarrollo de Aplicaciones Web

Brian Ducca

Índice

- Node Js
  - ¿Cómo funciona?
  - Callbacks
  - Programación orientada a eventos
  - Module.exports & require
- Express Js
  - Objeto Request
  - Objeto Response

Node Js

¿Qué es? ¿Por dónde empezar?

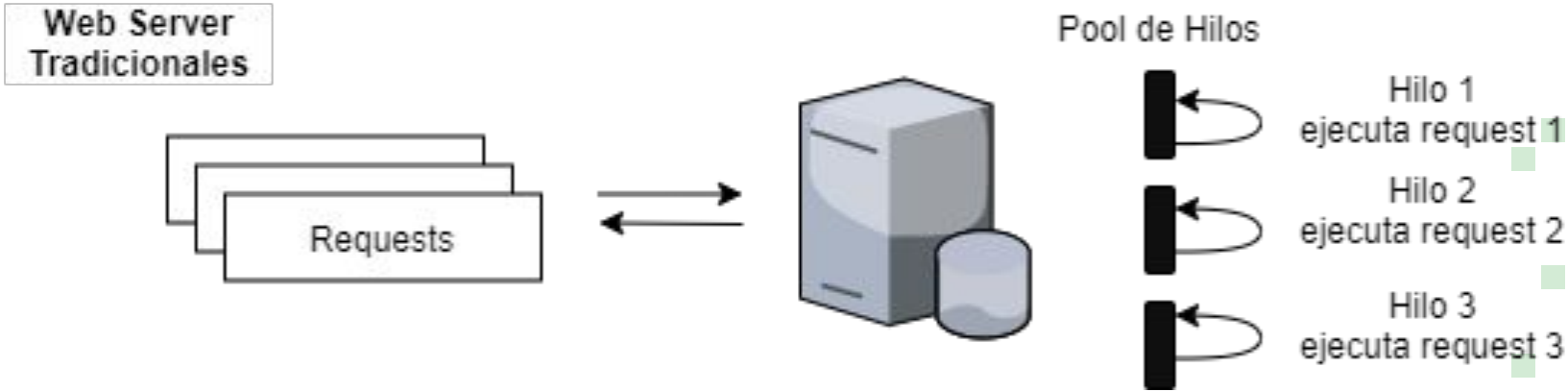
Node Js

- Orientado a eventos
- Trabaja sin bloquear el I/O (asíncrono)
- Sirve para
  - Aplicaciones Web
  - Api Rest
  - Chats Realtime
  - Aplicaciones de línea de comando

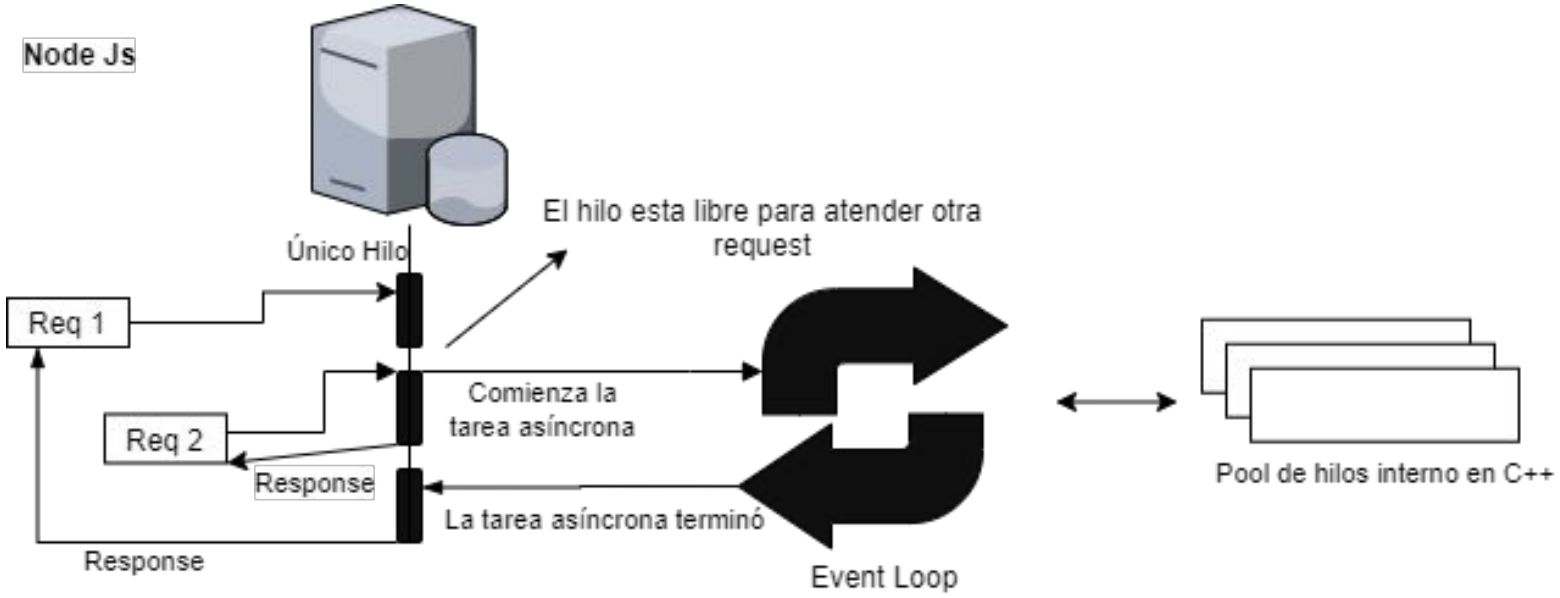
¿Cómo funciona?

- Único hilo
  - Consume menos recursos
- Trabaja de manera asincrónica
- Posee un “event loop”
  - Utiliza callbacks cuando la tarea termina

Web Server tradicionales vs Node



Web Server tradicionales vs Node



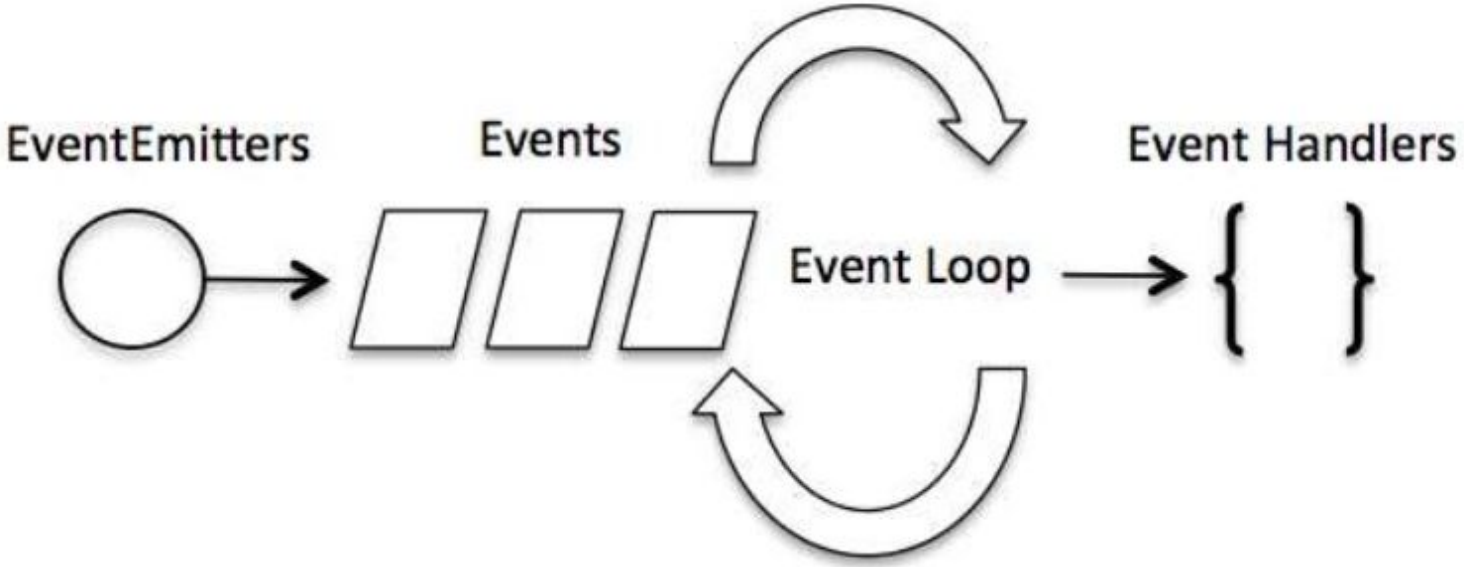
Callbacks

- Función que se ejecutará después de que otra función haya terminado de ejecutarse.

```
1  function hacerTarea(nombre, callback) {
2      alert(`Comenzando la tarea ${nombre}.`);
3      callback();
4  }
5  function tareaTerminada(){
6   alert('Fin de la tarea');
7  }
8  hacerTarea('Programación', tareaTerminada);
```

Programación Orientada a Eventos

- Esto junto con los callbacks dan soporte a la concurrencia de Node
- Loop principal que escucha eventos
  - Cuando se detecta alguno, son manejados por sus handlers
- Utilizan el patrón Observer


EventEmitters
Events
Event Loop
Event Handlers

Module.exports & require

- Module.exports exporta la porción de código que queremos tener disponible en otros módulos.

- Require es una función que va a importar todo lo exportado.

Module.exports & require

- En el archivo datos.js

```
module.exports = { nombre: 'Brian', apellido: 'Ducca' }
```

- En el archivo index.js

```
let persona = require('./datos.js');
console.log(persona.nombre + ' ' + persona.apellido);
```

Express Js

Métodos HTTP soportados por Express

- Get
- Post
- Put
- Delete

La definición de ruta tiene la siguiente estructura:

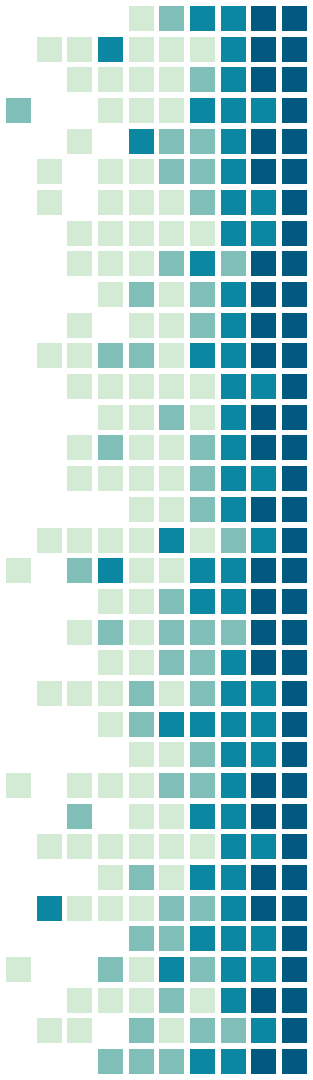
app.MÉTODO(ruta,handler)

Ejemplo

```
var express = require('express');
var app = express();

// GET method route
app.get('/', function(req, res) {
  res.send('hello world');
});

// POST method route
app.post('/', function (req, res) {
  res.send('POST request to the homepage');
});
```



Objeto Request

- Req.body
- Req.ip
- Req.params
- Req.secure

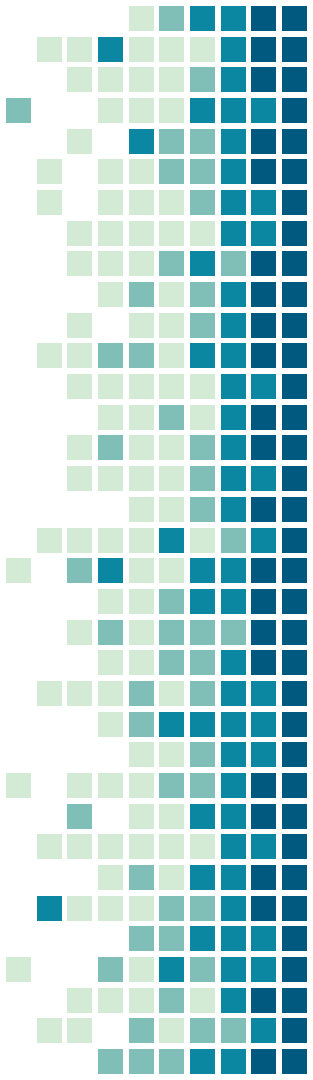
Ejemplo

```
var express = require('express’);

var app = express();

app.use(express.json());

app.post('/perfil, function (req, res, next) {
  console.log(req.body)
  res.json(req.body)
})
```



Objeto Response

- Representa la respuesta HTTP que Express envía cuando recibe una request HTTP.
- Ejemplo:

```
res.download('/reporteVentas.pdf', 'ventas.pdf');
res.json({ nombre: 'brian' , apellido:'ducca' });
res.redirect('http://www.google.com');
res.send({ usuario: 'bducca' });
```

METODO	DESCRIPCIÓN
res.download(path [, filename] [, options] [, fn])	Solicita un archivo para descargarlo.
res.end()	Finaliza el proceso de respuesta sin data.
res.json([body])	Envía una respuesta JSON.
res.jsonp()	Envía una respuesta JSON con soporte JSONP.
res.redirect()	Redirecciona una solicitud.
res.render()	Representa una plantilla de vista.
res.send([body])	Envía una respuesta de varios tipos. En el body puede ir un objeto Buffer, un String, un objeto o un Array por ejemplo.
res.status(code)	Setea el estado de la respuesta HTTP, permite encadenarse con los otros métodos.
res.sendStatus()	Establece el código de estado de la respuesta y envía su representación de serie como el cuerpo de respuesta.