UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

Diplomatura en programación web full stack con React JS

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 3: NodeJS: Introducción

Unidad 1: Fundamentos del lenguaje, instalación, npm, Express: creación de servidor

Presentación:

NodeJs es JavaScript “del lado del servidor”, en este módulo comenzamos el recorrido que nos llevará a convertirnos en desarrolladores backend!

Objetivos:

Que los participantes:

- Conozcan los fundamentos del lenguaje.
- Aprendan a instalar NodeJs.
- Sepan utilizar npm.
- Sean capaces de crear un servidor web.

Bloques temáticos:

1. Fundamentos del lenguaje
2. Instalación de NodeJS
3. Gestor de paquetes NPM
4. Introducción e instalación de Express
5. Inclusión de Express en una aplicación NodeJs
6. Concepto de servidor web
7. Ejemplo: creación de un servidor web
8. Trabajo Práctico

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

** El MEC es el modelo de E-learning colaborativo de nuestro Centro.*

Tomen nota:

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Fundamentos del lenguaje

¿Qué es NodeJS?

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. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.

Ventajas:

- Uso del mismo lenguaje en el cliente (JS) y en el servidor
- Reutilizacion de codigo (cliente y servidor)
- Usa el motor V8 de Chrome
- Entrada/Salida sin bloqueo
- Orientado a eventos
- Liviano
- Gran cantidad de paquetes

Uso del mismo lenguaje en el cliente y en el servidor

Una de las ventajas principales de NodeJS es la posibilidad de utilizar un único lenguaje de programación para hacer el desarrollo del frontend (JavaScript) y del backend (JavaScript en NodeJS). Lo cual reduce los tiempos de capacitación, disminuye las equivocaciones en sintaxis por cambio de lenguajes, y la posibilita tener un mayor manejo de toda la aplicación a desarrollar.

Reutilización de código

Al ser el mismo lenguaje que se utiliza en el cliente y el servidor (JavaScript), vamos a poder disponer de funciones que son aplicables (y código en general) que podremos utilizar tanto en el cliente de nuestra aplicación, como en el servidor. Esto reduce los tiempos de desarrollo, y mejora el mantenimiento de la aplicación.

Usa el motor V8 de Chrome

Es la tecnología sobre la cual corre NodeJS, la cual está ampliamente probada, es ágil y tiene varios años de desarrollo y madurez. Lo cual le da robustez, y velocidad a las aplicaciones desarrolladas en NodeJS.

Entrada/Salida sin bloqueo

Todas las operaciones que hacen uso de dispositivos (lectura en disco, consulta de información a través de Internet, etc) no bloquean a la aplicación, es decir, la aplicación sigue funcionando sin ningún tipo de ralentización. Este es uno de los principales motivos por los cuales NodeJS tiene una performance destacable frente a otras tecnologías para desarrollo de aplicaciones backend (servidoras).

Orientado a eventos

Todo el desarrollo en NodeJS está orientado a eventos que “disparan” una acción, lo cual hace que usemos el mismo concepto en toda la aplicación. Cuando desarrollamos en JavaScript, lado cliente, por ejemplo podemos tener un evento que es el onClick (cuando el usuario hace click), el cual “disparará” el código que nosotros especifiquemos. En NodeJS es similar, tenemos eventos que se “disparan” cuando el usuario hace una petición (como en la mayoría de las tecnologías server side), pero también se “disparan” eventos cuando se termina de leer una petición a otro equipo, cuando obtenemos un dato de una base de datos, etc.

Liviano

Uno de los puntos fuertes de NodeJS es lo “liviano” que es, esto quiere decir que con pocos requerimientos de Hardware podemos desplegar una aplicación en NodeJS. Muchas grandes empresas ya han comenzado a migrar sus desarrollos desde tecnologías más “pesadas” como JAVA/GRAILS/RUBY a NodeJS, y han tenido una considerable reducción en la cantidad de equipamiento necesaria para servir a sus clientes.

Gran cantidad de paquetes

Hoy en día existen muchas tecnologías para el desarrollo de aplicaciones, y una característica muy importante para elegir una tecnología es la cantidad de paquetes (bibliotecas, código reutilizable) que ya se encuentra desarrollado para la tecnología que se desea utilizar. NodeJS sobresale en este aspecto, ya que cuenta con miles de paquetes que realizan desde funciones muy sencillas hasta complejas, todo al alcance del desarrollador. Ya sea que se necesite de compilar código LESS a CSS, o que se necesite realizar operaciones complejas de autenticación de usuarios, seguramente existe un paquete en NodeJS que hace lo que estamos necesitando.

Casos de éxito

Linkedin

Las mejoras que experimentaron fueron sorprendentes para los mismos equipos de desarrollo. Antes de utilizar NodeJS utilizaban 15 servidores, con 15 instancias (servidores virtuales) en cada equipo físico. Con la incorporación de NodeJS redujeron la carga a solo 4 instancias que pueden manejar el doble de tráfico que antes.

O'Dell, J. (2011, August 16). Exclusive: How LinkedIn used Node.js and HTML5 to build a better, faster app. Recuperado de https://venturebeat.com/2011/08/16/linkedin-node

Walmart

NodeJS les permitió escalar muy bien sus aplicaciones, e integrar diferentes servicios.

O'Dell, J. (2011, August 16). Exclusive: How LinkedIn used Node.js and HTML5 to build a better, faster app. Retrieved from https://venturebeat.com/2011/08/16/linkedin-node

EBay

Comenzaron a utilizar NodeJS a causa que un grupo de ingenieros intentaron realizar una aplicación en JAVA y la misma necesitaba de muchos más recursos que los esperados. A partir de esta imposibilidad de seguir utilizando JAVA para su desarrollo, comenzaron a mirar hacia nuevas tecnologías, entre las cuales surgió la elección de NodeJS.

How We Built eBay's First Node.js Application. (n.d.). Retrieved from https://www.ebayinc.com/stories/blogs/tech/how-we-built-ebays-first-node-js-application/

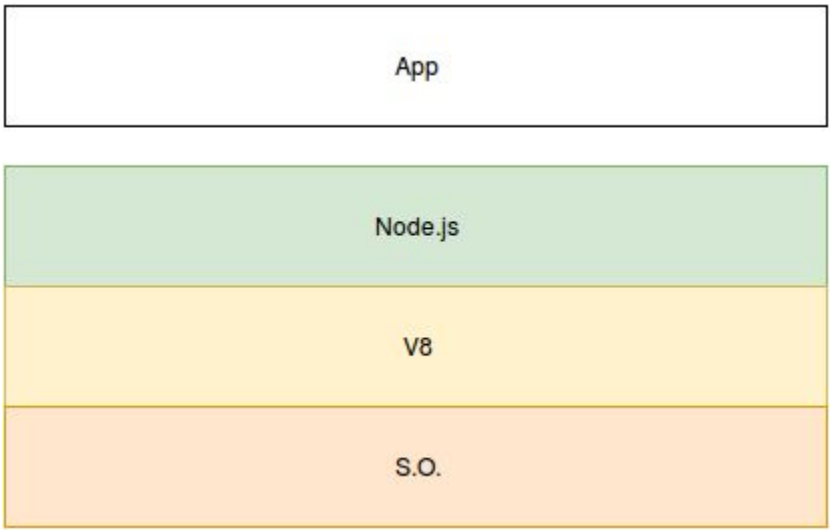
Conceptos básicos

Motor V8 de Chrome

- Es el motor de JavaScript de Google
- Compila JavaScript a código nativo
- Implementa ECMA-262
- Utilizado en Chromium y Node.js (entre otros productos)
- Está realizado en C++
- OpenSource
- Maneja automáticamente la asignación de memoria y garbage collection

El código fuente y documentación están disponibles en el sitio oficial de V8, https://developers.google.com/v8/

Stack (básico)



QUE SE NECESITA PARA CORRER UN SERVIDOR

- Reutilizar código
- Manejar archivos
- Manejar bases de datos
- Comunicarse a través de internet
- Poder aceptar peticiones y enviar respuestas
- Manejar operaciones que demoren

NodeJS y V8 son SingleThreaded (un único hilo) lo que presenta un problema para manejar operaciones de entrada/salida

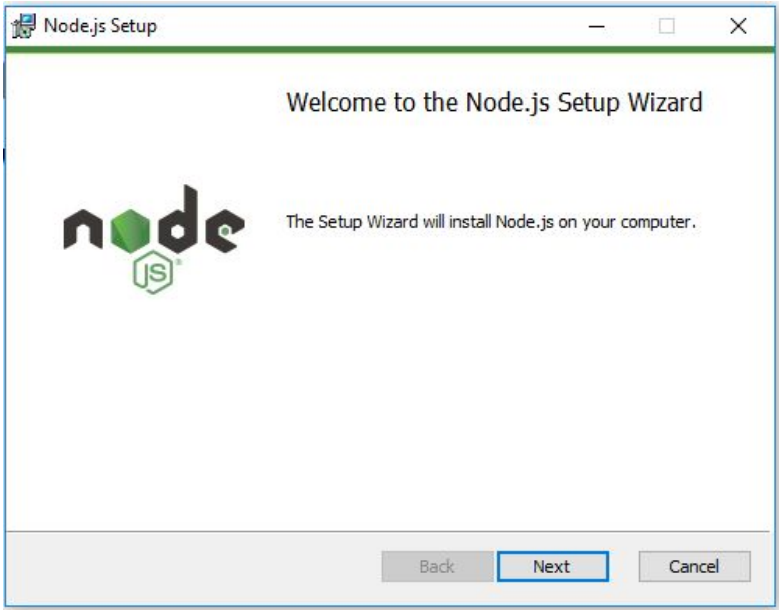
Es por esto que se añade Libuv que es una biblioteca que permite realizar procesos de entrada/salida (e/s) de manera asincrónica y simple.

libuv es una biblioteca de soporte multiplataforma con foco en las operaciones de E/S asincrónicas

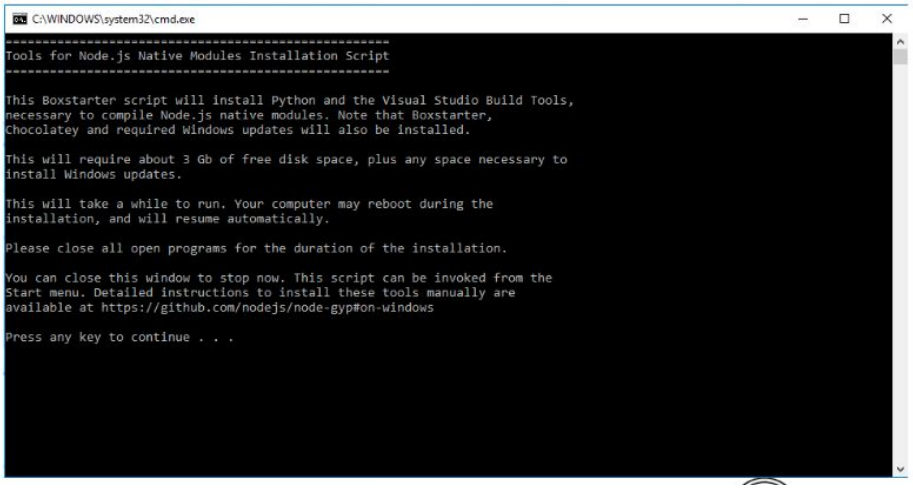
para mayor información sobre libuv http://libuv.org

2. Instalación de NodeJS

https://nodejs.org/es/download/



Al finalizar la instalación aparece esta pantalla



3. Gestor de paquetes NPM

Gestor de módulos para NodeJS. Se instala automáticamente con el NodeJS.

Para conocer la versión de node y npm que tenemos instalado (y así también saber si están instalados) usar desde la terminal, los siguientes comandos:

```
node -v
```

```
npm -v
```

Instalación de módulos con npm

Hay 2 formas de instalar módulos: local o globalmente

Localmente (recomendado)

El módulo deseado se instalará localmente en el proyecto que estemos trabajando, en una carpeta llamada node_modules.

```
$npm install [nombre_modulo] .
```

La carpeta node_modules se crea automáticamente al instalar un módulo.

Globalmente

Algunos módulos/aplicaciones, se pueden instalar para usarse desde cualquiera de nuestros proyectos.

```
$npm install -g [nombre_modulo] .
```

No es muy recomendable porque si actualizamos la versión del módulo/aplicación por un proyecto, estaremos afectando a todos los demás.

Ver la documentación de un módulo

(siempre que tenga el archivo .md creado):

```
$npm docs [nombre_modulo]  .
```

Se abre el navegador y va a la página de la documentación del módulo.

Utilizar los módulos

Desde el proyecto en Node.js

```
var modulo = require('modulo');  .
```

Package.json

NPM - package.json

Es un archivo fundamental para trabajar mejor y más fácil con npm.

Ventajas de usar package.json en nuestros proyectos

- No tenemos que instalar módulos uno a uno ya que se van a descargar de forma automática.
- Facilita la instalación de nuestra aplicación a otros desarrolladores.
- Todos los archivos y documentación de una determinada aplicación se almacena

en un solo lugar.

Se debe crear en el raiz de nuestro proyecto. La estructura quedará

app.js
package.json
node_modules

Estructura básica de package.json

```
{
        "name": "miapp",
        "version": "0.0.1",
        "dependencies": {
                "nombre_modulo": "version",
                "nombre_modulo2": "version"
        }
}
```

No es necesario instalar de a uno los diferentes módulos. Sólo se debe ejecutar por línea de comandos:

```
$npm install
```

Entonces:

1. NPM lee las dependencias incluídas en el archivo package.json
2. Instala automáticamente los módulos necesarios

A su vez, al compartir nuestra aplicación, no será necesario copiar la carpeta node_modules ya que se podrá generar automáticamente.

Creación de una aplicación

NPM - Correr una aplicación - Nivel básico

1. Crear la carpeta donde va a estar nuestro proyecto
2. Escribir en un archivo, el programa a ejecutar. Nombrarlo con extensión .js
3. En la consola ejecutar **node nombre_archivo.js** y oprimir la tecla Enter

1. Crear la carpeta donde va a estar nuestro proyecto
2. Ejecutar el comando `$npm init -f`

Esto creará en la carpeta, el archivo package.json que tendrá la siguiente forma:

```
{
  "name": "primerProyecto",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

3. Escribir en un archivo, el programa a ejecutar. Nombrarlo con extensión .js
4. Modificar package.json de la siguiente manera:

```
{
  "name": "primerProyecto",
  "version": "1.0.0",
  "description": "",
  "main": "primerPrograma.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node primerPrograma"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

NPM - Correr una aplicación - Utilizando package.json

5. En la consola, en la carpeta del proyecto, ejecutar el comando `npm start`

4. Introducción e instalación de Express

Express.js es un framework para Node.js que sirve para ayudarnos a crear aplicaciones web en menos tiempo ya que nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, y un largo etc…

Instalación

Suponiendo que ya está instalado nodejs y npm

```
> npm install --save express
```

5. Inclusión de Express en una aplicación NodeJs

```
'use strict'

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

6. Concepto de servidor web

En sitios web o aplicaciones web dinámicas, que accedan a bases de datos, el servidor espera a recibir peticiones HTTP del navegador (o cliente). Cuando se recibe una petición, la aplicación determina cuál es la acción adecuada correspondiente, de acuerdo a la estructura de la URL y a la información (opcional) indicada en la petición con los métodos POST o GET. Dependiendo de la acción a realizar, puede que se necesite leer o escribir en la base de datos, o realizar otras acciones necesarias para atender la petición correctamente. La aplicación ha de responder al navegador, normalmente, creando una página HTML dinámicamente para él, en la que se muestre la información pedida, usualmente dentro de un elemento específico para este fin, en una plantilla HTML.

Express posee métodos para especificar qué función ha de ser llamada dependiendo del verbo HTTP usado en la petición (GET, POST, SET, etc.) y la estructura de la URL ("ruta"). También tiene los métodos para especificar qué plantilla ("view") o gestor de visualización utilizar, donde están guardadas las plantillas de HTML que han de usarse y cómo generar la visualización adecuada para cada caso. El middleware de *Express*, puede usarse también para añadir funcionalidades para la gestión de cookies, sesiones y usuarios, mediante el uso de parámetros, en los métodos POST/GET. Puede utilizarse además cualquier sistema de trabajo con bases de datos, que sea soportado por *Node* (*Express* no especifica ningún método preferido para trabajar con bases de datos).

Más info en:
https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

7. Ejemplo: creación de un servidor web

```
'use strict'

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

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

8. Trabajo Práctico

Instalar NodeJS, instalar Express y transcribir el ejemplo aquí dado. Correrlo y verificar que funciona.

Bibliografía utilizada y sugerida

Express (n. d.) Recuperado de https://expressjs.com/es/

MDN - JavaScript (n.d.) Recuperado de:

https://developer.mozilla.org/es/docs/Web/JavaScript

NodeJs (n. d.) Recuperado de: https://nodejs.org/es/

NodeJS Documentacion (n. d.) Recuperado de https://nodejs.org/es/docs/

NPM (n. d.) Recuperado de https://www.npmjs.com/package/page

Wikipedia - NodeJS (n. d.) Recuperado de https://es.wikipedia.org/wiki/Node.js

Lo que vimos:

Introducción a NodeJs y Express.

Lo que viene:

Express: ruteo, recepción de formularios.